Hes·so VALAIS WALLIS

Haute Ecole de Gestion
Hochschule für Wirtschaft

Filière Informatique de gestion
Studiengang Wirtschaftsinformatik

Travail de Bachelor 2022

Technologies de l'information et de la communication (TIC)

Vers une identité électronique autogérée

Auteur :

Valon Rexhepi

Professeur :

Jean-Luc Beuchat

HES-SO Valais-Wallis • rte de la Plaine 2 • CP 80 • 3960 Sierre
+41 58 606 93 11 • info.informatique@hevs.ch • www.hevs.ch/ig

swissuniversities

EQUAL-SALARY CERTIFIED

Valais excellence

Résumé

Dans une époque où le ***cybercrime*** fait de plus en plus de dégâts, il devient primordial de trouver une solution pour ralentir cette menace. Les propositions mises en place actuellement, comme la combinaison de l'identifiant avec un mot de passe, sont souvent aussi la porte d'entrée des cybercriminels. L'idée nouvelle de l'identité autogérée se présente alors comme une solution prometteuse à la résolution de cette problématique. Cette identité autogérée, c'est aussi une des idées retenues par le Conseil fédéral pour la mise en place d'un ***e-ID*** étatique suisse.

Tout au long de ce document, nous allons découvrir ensemble ce qu'est l'identité autogérée. Nous commencerons par aborder les blocs théoriques qui forment son écosystème, puis nous nous pencherons sur des solutions existantes proposant la mise en place de cette technologie. Ensuite, nous aborderons certains défis émis par le *Département fédéral de justice et police*, pour la création de l'***e-ID*** étatique, en y proposant des solutions et en présentant leurs limites. Après cela, nous présenterons différentes preuves de concept démontrant les possibilités d'une solution existante. Nous synthétiserons les éléments rencontrés lors de l'élaboration de ce travail, nous présenterons également les limites de celui-ci et finalement nous ouvrirons la voie à de nouvelles perspectives de recherches.

Mots clés : Self-Sovereign Identity (SSI), blockchain, identité numérique (e-ID)

Avant-propos et remerciements

Au début de l'année 2021, le peuple suisse rejetait la loi proposée sur l'identité électronique. Ce refus a conduit le Conseil fédéral à demander au Département fédéral de justice et police d'émettre un document présentant de nouvelles solutions pour la mise en place d'un ***e-ID***. À la fin de la même année, une solution retenue était celle de l'identité autogérée, ou en anglais *Self-Sovereign Identity*.

Dans ce travail, nous nous efforcerons de présenter ce qu'est l'identité autogérée, son fonctionnement, mais également ses limites. Nous répondrons aussi à diverses questions en suspens posées par le Département fédéral de justice et police grâce aux connaissances acquises lors de la présentation de la *Self-Sovereign Identity*. Enfin, nous ouvrirons la discussion sur l'avenir de l'***e-ID*** en Suisse. Tout au long de ce document, nous mettrons un accent sur la vulgarisation des différents thèmes pour que ceux-ci soient accessibles à tous. Pour ce faire, nous utiliserons une démarche de recherches rigoureuses. La difficulté principale rencontrée lors de l'élaboration de ce travail fût la compréhension des différentes technologies et notion entourant l'identité autogérée.

Nous concluons cet avant-propos en remerciant les différentes personnes qui ont amené leur aide pendant la réalisation de ce travail.

Monsieur Jean-Luc Beuchat, le professeur responsable, pour son suivi tout au long de ce travail et pour nous avoir permis d'étendre nos connaissances dans des domaines encore peu développés. Nous le remercions également pour les idées, relectures et corrections qu'il a apportées à ce projet.

Monsieur Fabian Cretton et Monsieur Antoine Induni pour l'échange de ressources et la collaboration sur lesquelles nous avons pu nous reposer pour concevoir ce document.

Madame Maruschka Bussien pour son soutien moral et pour les nombreuses relectures de ce bachelor.

Table des matières

Table des figures

Liste des tableaux

1 Introduction

1.1 Contexte

Le travail suivant s'inscrit tout d'abord dans un contexte académique. En effet, dans le cadre de la formation en Informatique de Gestion dispensée par la Haute École Spécialisée de Suisse Occidentale, les étudiants en dernière année se voient attribuer un thème d'étude, selon leurs préférences, pour réaliser leur travail de bachelor. Le thème de ce travail, intitulé *Vers une identité électronique autogérée*, a été proposé et encadré par Jean-Luc Beuchat.

Ce travail s'inscrit également dans un contexte politique suisse. Le 7 mars 2021, le peuple rejetait la loi sur l'identité électronique [1]. L'argument principal contre cette loi reposait sur le fait que des entreprises privées fournissaient les services de gestion de l'identité électronique, celles-ci pouvant alors collecter des données, comme l'historique des authentifications, sur les citoyens et les stocker de manière centralisée. Le 10 mars 2021, plusieurs motions au même libellé sont soumises, et précisent que seules les informations nécessaires doivent être collectées et stockées de manière décentralisée. Ainsi le Conseil fédéral a demandé au Département fédéral de justice et police d'émettre un texte présentant plusieurs solutions pour répondre à la problématique. À la suite de la publication de ce document le 2 septembre 2021, une consultation publique est ouverte, et l'une des solutions identifiées pendant ce processus est celle de la *Self-Sovereign Identity* [2].

Ce travail représente ainsi une base d'analyse et de compréhension sur l'identité autogérée. Il est réalisé entre le 5 mai et le 29 juillet 2022. Le 29 juin 2022, une nouvelle proposition d'initiative sur l'***e-ID*** a été mise en consultation jusqu'en octobre 2022 (RTS, 2022).

1. La loi intitulée "Identité électronique : la loi sur l'e-ID" a été rejetée à plus de 60% (DÉPARTEMENT FÉDÉRAL DE JUSTICE ET POLICE, 2021).
2. En français, elle est nommée *"identité autogérée"*.

1.2 Méthodologie

Pour effectuer ce travail, une recherche approfondie des ressources à disposition est effectuée. La thématique est récente, impliquant donc une limitation des produits et solutions disponibles pour la création d'une identité autogérée. Ils constituent pour l'instant un marché de niche, autant du côté client que développeur. Les ressources utilisées sont constituées de travaux scientifiques, de vidéos, d'interviews et d'ouvrages relatifs à l'identité autogérée mais également de ressources techniques proposées par les acteurs du domaine. Notons également que les différentes spécifications techniques proposées par le ***World Wide Web Consortium*** doivent être attentivement suivies. En effet, celles-ci pourraient devenir des standards reconnus à la mise en place de solutions pour l'identité autogérée.

Ce travail est donc plus orienté sur la recherche que la production. De ce fait, l'application complète d'une méthodologie ***Agile*** se révèle contre-productive. Toutefois, le choix de mener des réunions hebdomadaires avec les membres de l'équipe de recherche sur l'identité autogérée et de maintenir un suivi des tâches du projet s'est mis en place naturellement. Ces réunions servent à mettre en commun les découvertes de la semaine sur la thématique, l'environnement qui en découle et pour discuter des problèmes potentiellement rencontrés, que cela relève de l'incompréhension de notions ou de mise en place technique d'éléments d'exemple aux solutions disponibles.

L'auteur met également en place un journal de bord ainsi que des comptes-rendus personnels pour le suivi des activités journalières et des dates butoirs. Ces documents permettent d'avoir une vue générale du travail réalisé et à réaliser tout au long du projet. L'application de cette méthode permet une meilleure gestion du temps, une amélioration de la communication au sein du groupe et un renforcement du partage des connaissances.

1.3 Sujet traité

La thématique de l'identité autogérée est vaste. Chaque bloc constituant celle-ci peut servir comme base pour réaliser des documents complexes et complets sur le sujet. Même si l'identité autogérée est une idée nouvelle, la majorité des éléments qui la constituent sont connus depuis déjà plusieurs années. La complexité de certaines notions peut nécessiter plusieurs semaines de recherches pour leur compréhension.

Dans ce travail, nous nous efforçons de vulgariser les notions relatives à l'identité autogérée et à l'environnement qui l'entoure pour que celles-ci puissent être comprises de tous, et que chacun puisse imaginer les changements qu'une telle technologie engendre sur la société. Cette compréhension est, de plus, particulièrement importante compte tenu du débat politique suisse autour de l'identité électronique.

Cette situation politique mène également à une discussion sur les défis établis par le Département fédéral de justice et police pour déployer une identité autogérée en Suisse.

1.4 Objectifs

Ce document est séparé en trois parties principales. Premièrement, l'état de l'art traite :

1. de l'identité autogérée en expliquant l'idée, son fonctionnement et les blocs qui la constituent ;
2. de la *Distributed Ledger Technology*[3], en expliquant ce qu'elle est, son utilité avec l'identité autogérée et en donnant des pistes de recommandation sur la technologie la plus adaptée pour répondre à la problématique ;
3. des différentes solutions d'identité autogérée existantes en exposant leurs caractéristiques principales, leurs avantanges et leurs inconvénients.

Deuxièmement, nous traiterons l'objectif principal de ce travail qui est l'apport de réponses aux différents défis établis par le Département fédéral de justice et police. Pour ce faire, nous analyserons les défis posés et y répondrons grâce aux connaissances établies dans l'état de l'art et en prenant position sur ceux-ci.

Finalement, nous exposerons des preuves de concept sur la réalisation de certains composants pour la mise en place d'une identité autogérée. Nous analyserons ensuite les résultats obtenus avec la mise en place de ces concepts.

3. En français, le terme se traduit par *"technologie de registre distribué"*.

2 | État de l'art

Dans ce chapitre, nous nous concentrons sur l'état de l'art actuel. Le terme anglais utilisé pour l'identité autogérée est la *Self-Sovereign Identity (SSI)*. Nous utilisons désormais cette appellation pour la suite de ce document. Nous commençons par parler de la *SSI*, puis nous poursuivons avec la *Distributed Ledger Technology (DLT)* et nous concluons en présentant les différentes solutions qui existent déjà sur le marché.

2.1 Identité autogérée

Dans cette section, nous découvrons ensemble ce qu'est la *SSI*. Pour ce faire, nous devons d'abord comprendre la problématique majeure de l'Internet actuel. Nous discutons ensuite des différents modèles d'identités numériques utilisés. Grâce à ces connaissances, nous pouvons présenter l'identité autogérée et son fonctionnement. Puis, nous observons un exemple d'utilisation de la technologie. Finalement, nous analysons les bénéfices et les défis qui entourent la *SSI*.

2.1.1 Problématique de l'Internet actuel

Un peu d'histoire

Pour pouvoir comprendre ce qui a conduit à la nécessité d'avoir une *SSI*, il nous faut tout d'abord faire un saut dans le passé. C'est entre le début des années 60 et la fin des années 90 qu'Internet voit le jour. Chaque décennie marque alors un tournant dans la création de celui-ci.

Dans les années 60, l'informaticien Joseph Carl Robnett Licklider partage alors l'idée d'un réseau global d'ordinateurs avec l'un de ses collègues du département de la *Defense Advanced Research Projects Agency*, l'agence pour les projets de recherche avancée du département de la Défense des États-Unis. L'idée devient alors une réalité en 1969, quand des ordinateurs universitaires américains se connectent sur un réseau commun. L'*Advanced Research Projects Agency Network*, le réseau de l'agence pour les projets de recherche avancée, est né (THOMAS JEFFERSON UNIVERSITY, 2016).

Dans les années 70, l'*ARPANET* continue de s'agrandir. La nécessité de connecter plusieurs réseaux entre eux devient alors de plus en plus problématique. En 1973, les scientifiques Robert Elliot Kahn et Vinton Gray Cerf travaillent alors sur le développement d'un protocole qui servira de solution à ce problème. Ce protocole est le *Transmission Control Protocol/Internet Protocol (TCP/IP)* (THOMAS JEFFERSON UNIVERSITY, 2016). Il permet la communication entre ordinateurs sur un réseau, par exemple Internet. Pour ce faire, il sépare les données en paquets avant de les envoyer à un destinataire.

Dans les années 80, le défi de connecter les réseaux entre eux est réussi, mais l'infrastructure à mettre en place pour y parvenir coûte cher. Le but devient alors de réussir à les connecter à travers un réseau bon marché. C'est en 1982 que le système PhoneNet [1] voit le jour et est connecté à *ARPANET* ainsi qu'au tout premier réseau commercial. C'est aussi pendant cette décennie que le *Domain Name System (DNS)* voit le jour [2] (THOMAS JEFFERSON UNIVERSITY, 2016).

Enfin, dans les années 90, l'*Advanced Research Projects Agency Network* est déclassé par le département de la Défense des États-Unis. C'est alors au *Conseil européen pour la recherche nucléaire (CERN)* que Tim Berners-Lee et ses collègues développent l'***Hypertext Markup Language*** ainsi que l'***Uniform Resource Locator*** et donnent ainsi naissance au ***World Wide Web***.

La problématique

Il est important que nous nous arrêtions un moment sur l'un des protocoles qui compose le *TCP/IP*, le Protocole Internet [3]. Il définit comment transmettre ces données à travers le réseau en joignant des informations supplémentaires aux paquets. Une de ces informations est l'adresse *IP*. Chaque ordinateur connecté à Internet possède une adresse *IP*. Cette adresse, comme pour une adresse postale, va permettre de connaître le destinataire que nous voulons contacter sur le réseau (IBM, 2022). Cette adresse et ce protocole vont permettre de localiser une machine et de communiquer avec celle-ci. Cependant, l'une des limites est que nous ne disposons d'aucune autre information sur ce qui se trouve "derrière" cette machine, il est donc impossible de savoir avec qui ou quoi nous communiquons (PREUKSCHAT et REED, 2021). C'est ici que se trouve le plus gros point faible de l'Internet actuel : il n'y a aucune couche d'identité.

Dans la vie réelle, si nous souhaitons par exemple nous inscrire dans une école, l'établissement peut nous demander de lui fournir une carte d'identité et même de nous rencontrer. Cela permet de créer un lien de confiance avec l'établissement en prouvant que nous sommes bien la personne que nous prétendons être. Sur Internet, c'est une autre histoire. Il est impossible pour un site internet de rencontrer physiquement une à une les personnes qui s'y connectent pour vérifier leur identité avant de les laisser accéder à leur compte.

Dans le livre PREUKSCHAT et REED, 2021, nous pouvons lire une prédiction faite par Kim Cameron en 2005, ici traduite en français *"Si nous ne faisons rien, nous serons confrontés à une prolifération rapide d'épisodes de vol et de tromperie qui réduiront de manière cumulative la confiance du public dans l'Internet."*. Plus de 15 ans plus tard, nous ne pouvons que constater la justesse de cette prédiction.

1. La connexion se fait grâce aux lignes téléphoniques.
2. Il va permettre de traduire des noms de domaine Internet en adresse IP.
3. En anglais, il se nomme *"Internet Protocol (IP)"*.

Pour combattre le manque de cette couche d'identité sur Internet, de nombreuses solutions ont été proposées et adoptées à travers le temps. Le ***Login***, avec l'identifiant et mot de passe, est l'une de ces solutions (PREUKSCHAT et REED, 2021). La capacité de ces solutions, à répondre au manque d'une couche d'identité, semble cependant limitée.

La problématique en chiffre

Dans la société actuelle où la digitalisation est de plus en plus présente, il n'est pas étonnant qu'en 2015 déjà, Ginni Rometty (la PDG d'IBM) disait *"Le **cybercrime** est, par définition, la plus grande menace à chaque profession, chaque industrie, chaque entreprise dans le monde."* (MORGAN, 2015).

Pour donner quelques chiffres, il est estimé qu'en 2025, le coût des dégâts causés par le **cybercrime** se montera à 10.5 trillons de dollars annuellement. Dans les 9 premiers mois de 2021, il y a eu plus de 495 millions d'attaques au ransomware[4]. Aux États-Unis, ces attaques sont évaluées à plus de 623.7 millions de dollars en 2021. Dans la majorité des cas, ce sont des mots de passe trop faibles qui sont la cause de ces attaques (ZAHARIA, 2022). En Suisse, ces attaques continuent d'augmenter chaque année (CENTRE NATIONAL POUR LA CYBERSÉCURITÉ, 2020). En 2019, les États-Unis ont été touché par une attaque de grande envergure qui endommagea l'infrastructure de plus de 700 fournisseurs de soins de santé. Cette attaque eu pour conséquence une incapacité des services hospitaliers à fournir leur prestations, les forçant ainsi à diriger certains patients vers d'autres hôpitaux non impactés par l'attaque (EMSISOFT MALWARE LAB, 2019).

Nous pouvons donc nous rendre compte que le manque d'une couche d'identité sur Internet représente non seulement une menace économique. mais également une menace vers la société elle-même du fait de sa digitalisation croissante. Avec l'ajout de celle-ci, certaines attaques deviennent très difficiles à réaliser car le criminel devrait réussir à falsifier son identité. La couche d'identité serait alors une des solutions permettant la réduction des risques liés aux ***cybercrimes***.

Réseaux de nouvelles générations

Pour répondre à ces différentes problématiques, les chercheurs se creusent la tête depuis des années pour trouver des solutions d'amélioration à la sécurité sur Internet. L'une d'entre elle est la *SSI*, dont nous parlerons dans les prochaines sections de ce chapitre. Une autre solution est *Scalability, Control, and Isolation On Next-Generation Networks (SCION)*[5]. *SCION* est une architecture de réseau proposé par des chercheurs de l'École polytechnique fédérale de Zürich. Cette solution se présente comme étant d'une sécurité élevée, mais également de haute efficacité de transmission des paquets (SCION, s. d.). *Swisscom*, le leader du marché de la télécommunication en Suisse, propose déjà *SCION* comme prestation à ses clients (SWISSCOM, s. d.).

4. En français, le terme se traduit par *"logiciel d'extorsion"*.
5. En français, le nom complet est *"extensibilité, contrôle et isolation sur les réseaux de prochaine génération"*.

2.1.2 Modèles d'identités numériques

Pour comprendre l'impact de la *Self-Sovereign Identity*, l'étude des différents modèles d'identités numériques est nécessaire. Nous allons donc analyser trois modèles différents. Pour ce faire, nous nous appuyons sur l'article (RUFF, 2018) qui propose ces trois modèles ainsi que sur le livre (PREUKSCHAT et REED, 2021) qui reprend les définitions de l'article en y ajoutant de nouvelles informations.

Le modèle traditionnel

Le modèle traditionnel, aussi appelé *modèle centralisé*, *modèle en silo* ou encore *identité basée sur le compte*, est le plus ancien ainsi que le plus utilisé aujourd'hui. Dans ce modèle, une organisation nous permet de créer, ou nous donne directement, un identifiant numérique pour nous représenter. Cet identifiant nous sert alors de clé d'accès aux services de l'organisation (RUFF, 2018). Prenons deux exemples concrets.

Dans le premier, nous souhaitons effectuer une formation dans une école quelconque. Nous devons alors nous inscrire à celle-ci via le formulaire d'inscription mis à disposition sur le site internet de l'école. Pendant l'inscription, il nous est également demandé de fournir des documents pour prouver notre identité, comme une copie de la carte d'identité ou une attestation de résidence. Quelques jours après notre inscription, nous recevons par courrier postal une confirmation, un livret d'informations, ainsi qu'une carte d'étudiant. Dans le livret d'information, nous apprenons que cette carte nous permet d'ouvrir les portes du bâtiment de l'école ou encore d'utiliser les photocopieuses et imprimantes de celle-ci.

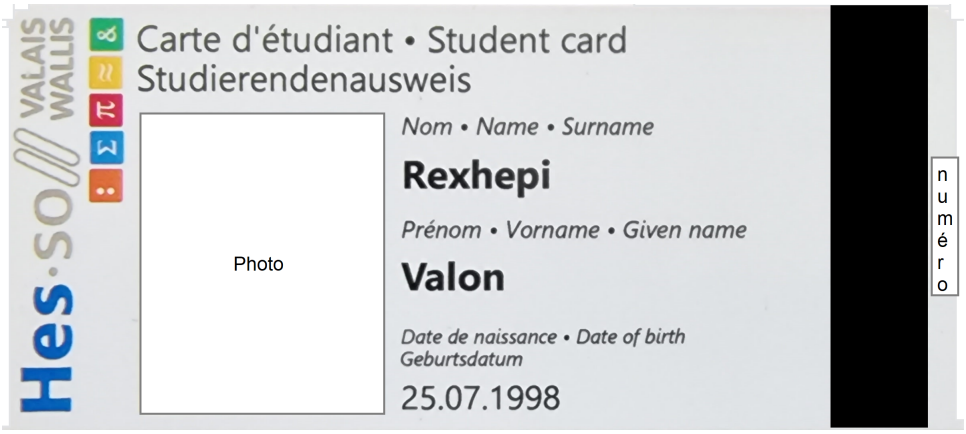


FIGURE 2.1 – *Exemple de carte d'étudiant*
Source: *de l'auteur*

Dans le deuxième exemple, nous souhaitons faire une commande de divers accessoires bureautiques sur le site internet *Amazon*[6]. Pour ce faire, il nous est demandé de créer un compte. Nous devons donc donner notre nom complet, choisir un nom d'utilisateur (représenté par un numéro de téléphone ou une adresse E-mail), ainsi qu'un mot de passe constitué d'au minimum six caractères. Ces informations serviront ensuite pour nous représenter sur les services d'*Amazon*.

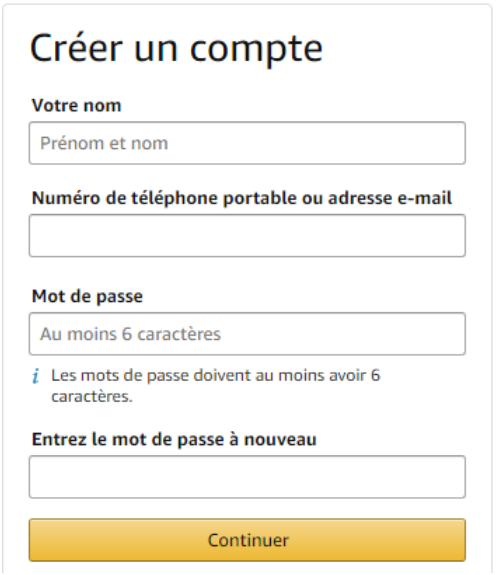


FIGURE 2.2 – *Exemple de création de compte sur Amazon*
Source: *de l'auteur*

Dans les deux cas, le schéma de fonctionnement peut être représenté selon la figure 2.3.

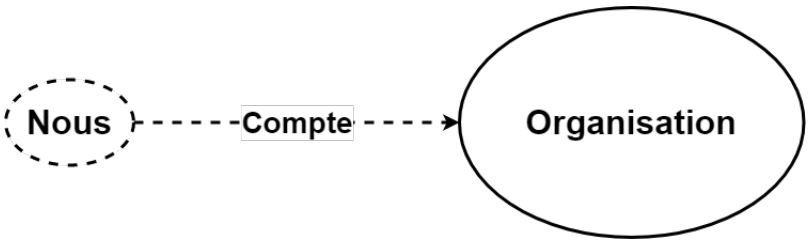


FIGURE 2.3 – *Fonctionnement du modèle traditionnel*
Source: *de l'auteur inspirée par* RUFF*, 2018*

Ainsi, c'est nous qui allons créer un compte pour pouvoir nous lier à l'organisation. Cela va nous permettre d'être représenté en tant qu'individu et ainsi avoir accès à un panel de service personnalisé[7]. Dans ce modèle, la confiance entre nous et l'organisation va prendre la forme d'un secret partagé, par exemple la paire *nom d'utilisateur - mot de passe* ou encore la reconnaissance d'empreinte (RUFF, 2018).

6. *Amazon.com* est un site de commerce en ligne.
7. Sur *Amazon.com*, il est par exemple possible de consulter un historique des commandes que nous avons effectuées.

Les avantages de ce modèle, selon RUFF, 2018, sont les suivants :

1. il est énormément utilisé, son fonctionnement est compris et est simple à mettre en place ;
2. il aide l'organisation à réduire les risques liés à la gestion des données du client en gardant celles-ci en interne ;
3. il permet d'avoir un identifiant unique pour chaque relation, tant que celui-ci n'est pas réutilisé, par exemple en utilisant la même paire *nom d'utilisateur - mot de passe* sur plusieurs applications différentes.

Les inconvénients de ce modèle, selon RUFF, 2018, sont les suivants :

1. une rupture dans la sécurité informatique d'une organisation peut entraîner des conséquences dramatiques, comme l'exposition publique d'informations personnelles ;
2. l'expérience utilisateur n'est pas optimale car il force l'utilisateur à se souvenir de tous ses identifiants, sauf si nous utilisons un ***Password Safe*** ;
3. chaque organisation doit devenir experte en sécurité informatique pour assurer la protection des données des utilisateurs. C'est également le cas dans les autres modèles car il y a des données, autres que celles des utilisateurs, à protéger.

Le livre PREUKSCHAT et REED, 2021 ajoute également que toutes les données que nous partageons avec l'organisation ne nous appartiennent plus et sont hors de notre contrôle. L'entreprise devrait ainsi collecter uniquement les informations nécessaires au fonctionnement de son affaire.

Le modèle fédéré

Le modèle fédéré, ou modèle de relation par tierce partie[8], est de plus en plus utilisé aujourd'hui. Dans ce modèle, nous ne créons pas directement un identifiant avec l'organisation mais nous passons par ce qui s'appelle un *fournisseur d'identité*, en anglais *identity provider*. La figure 2.4 représente le fonctionnement du modèle.

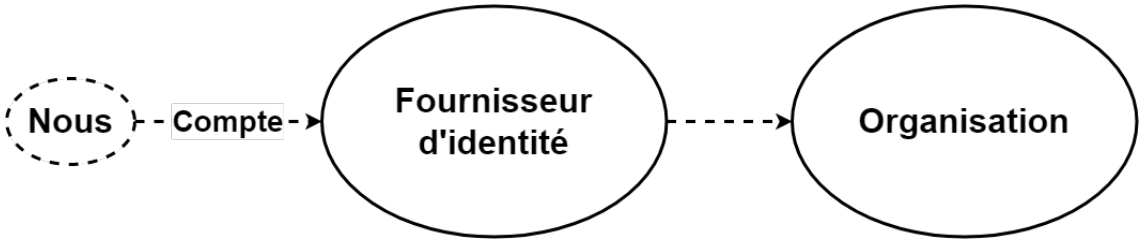


FIGURE 2.4 – *Fonctionnement du modèle fédéré*
Source: *de l'auteur inspirée par* RUFF*, 2018*

8. Le nom anglais est *"identity provider relationship model"*. Traduit littéralement, cela signifie *modèle de relation par fournisseur d'identité*.

Ainsi, il est possible de créer une seule identité avec un *identity provider* et celui-ci va s'occuper de nous connecter et/ou de partager des informations avec les organisations (PREUKSCHAT et REED, 2021). Il y a de nombreux protocoles qui ont été développés pour mettre en place ce modèle, comme le *Security Assertion Markup Language* ou encore l'*OpenID Connect*. Ils permettent tous de faire du *single sign-on*[9].

Sur de nombreux sites internet, il est de plus en plus courant de voir des boutons de connexion *Facebook*, *Google* ou *Microsoft* qui vont utiliser ces protocoles pour nous permettre de nous identifier. Le site représente alors l'organisation tandis que le fournisseur sur le bouton, par exemple *Facebook*, représente l'*identity provider* (PREUKSCHAT et REED, 2021).

FIGURE 2.5 – *Exemple de boutons de connexion*
Source: *de l'auteur*

L'avantage de ce modèle, selon RUFF, 2018, est le fait que celui-ci améliore l'expérience utilisateur en permettant une authentification simplifiée et centralisée réduisant ainsi le nombre d'identifiants à retenir.

Les inconvénients de ce modèle, selon PREUKSCHAT et REED, 2021, sont les suivants :

1. il n'y a pas un fournisseur d'identité qui fonctionne sur tous les services disponibles. Le problème, qui était jusque là de se souvenir de ses identifiants, passe alors à se souvenir de son fournisseur d'identité et des identifiants utilisés sur celui-ci ;
2. il peut être désagréable d'avoir une entité entre nous et l'organisation qui se voit alors capable de "surveiller" les activités de connexion, ce qui est le cas actuellement avec les cartes de crédit ;
3. plus le fournisseur d'identité est gros, plus l'infraction de celui-ci peut "rapporter gros" pour les cybercriminels ;
4. les comptes créés sur des *identity provider* ne sont pas portables, ainsi si nous supprimons notre compte *Facebook* par exemple, toutes les connexions établies grâce à cet identifiant seront perdues ;
5. le fournisseur d'identité ne se trouve pas en position pour assurer le transfert sécurisé d'informations sensibles. En effet, celui-ci est potentiellement la cible d'attaques informatiques mais peut également tromper l'utilisateur sur ses motivations véritables[10].

9. En français, c'est l'*"authentification unique"*.
10. Comment être sûr que cette tierce partie ne dérobe pas nos informations ?

Le modèle décentralisé

Un nouveau modèle a fait récemment son apparition, c'est le modèle décentralisé. Ce modèle représente une relation bilatérale entre nous et l'organisation. La figure 2.6 représente le fonctionnement du modèle.

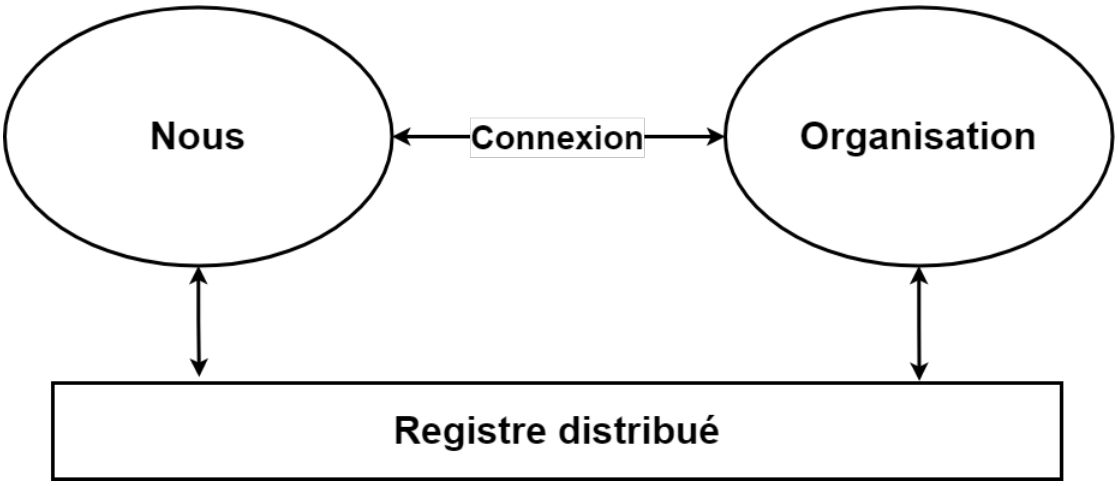


FIGURE 2.6 – *Fonctionnement du modèle décentralisé*
Source: *de l'auteur inspirée par* RUFF*, 2018*

La différence majeure entre ce modèle et les autres est qu'il n'est pas basé sur un compte avec un identifiant mais sur une relation bidirectionnelle entre l'organisation et nous. Cela signifie qu'aucune des deux entités n'est plus "forte" que l'autre. Rien, ni personne, ne va contrôler, fournir ou posséder la relation. Ainsi, le concept de compte est remplacé par celui d'une connexion. Pour que celle-ci persiste, les deux entités doivent continuer de la maintenir (PREUKSCHAT et REED, 2021). Ce type de connexion est appelée ***peer-to-peer***. Chaque entité dans la connexion représente un *pair* et elle est directement établie entre les différents *pairs*, sans l'intervention d'une partie tierce.

Une analogie fournie par l'ouvrage PREUKSCHAT et REED, 2021 pour représenter le fonctionnement du modèle décentralisé, est un rapprochement avec le monde réel. En effet, dans la vie de tous les jours, pour prouver notre identité, nous sortons notre portefeuille et présentons les différents certificats nécessaires que nous avons obtenus par diverses parties. Par exemple, si nous devons montrer notre permis de conduire à la police, celui-ci nous a été donné par le service responsable de la circulation routière. Avec ce modèle, c'est la même chose, mais les certificats, le portefeuille et les connexions sont digitales.

Nous verrons plus en détail le fonctionnement de ce modèle, ainsi que ses avantages et les défis qui l'entourent, dans les sections suivantes de ce chapitre. Car ils sont directement liés au fonctionnement de la *Self-Sovereign Identity*. Notons également que dans le chapitre 2.2.3, nous parlons et expliquons la ***Public Key Infrastructure*** qui est également un type de modèle décentralisé.

2.1.3 Présentation

L'évolution du modèle d'identité numérique est le résultat de la recherche de la satisfaction de trois contraintes primordiales (TOBIN et REED, 2016) :

1. la sécurité, l'identité doit être protégée contre le vol ou l'exposition d'informations ;
2. le contrôle, celui qui possède l'identité doit avoir un contrôle complet sur ce qu'il souhaite partager de son identité et avec qui. Ce contrôle peut cependant être révoqué, par exemple si le permis de conduire d'une personne est retiré, celle-ci ne peut plus le contrôler ;
3. la portabilité, l'identité doit pouvoir être réutilisée et non unique à chaque service.

Comme nous l'avons vu précédemment, l'évolution la plus récente et qui semble remplir au mieux ces contraintes est la *SSI*. Sa caractéristique de n'impliquer aucune partie tierce dans fonctionnement permet un contrôle complet de l'identité par celui qui la possède. Le terme *Self-Sovereign Identity* peut d'ailleurs se traduire en français par "*L'identité d'une personne qui ne dépend ni n'est soumise à aucun autre pouvoir ou état*" (PREUKSCHAT et REED, 2021).

Il y a deux fausses propriétés répandues à propos de la *SSI*. La première avance que le détenteur de l'identité est le seul à pouvoir faire des affirmations sur lui-même. Bien évidemment, comme dans la vraie vie, ce n'est pas le cas. N'importe quel organisme auquel le détenteur fait confiance peut faire des affirmations sur son identité. Il suffit de jeter un coup d'oeil dans nos portefeuilles pour s'en rendre compte. Que ce soit carte d'identité de l'état, carte de membre de salle de sports, carte d'étudiant et bien d'autres. C'est la même chose pour la *SSI*. La deuxième avance que la *SSI* est seulement pour représenter les personnes, mais ce n'est pas le cas. Que ce soit une organisation, un objet, un animal, n'importe quelle entité qui nécessite une identité sur Internet peut utiliser le modèle du *SSI* pour être représentée [11] (PREUKSCHAT et REED, 2021).

La meilleure façon de se représenter ce qu'est la *SSI* est de l'imaginer comme un portefeuille électronique, dans lequel nous pouvons ajouter des informations sur notre identité ou demander à d'autres entités de le faire (TOBIN et REED, 2016).

Il est important de comprendre que pour que la *SSI* soit réellement *autogérée*, l'infrastructure de celle-ci doit se trouver dans un environnement de confiance et indépendant. Cet environnement n'appartient à personne et de ce fait nul n'a l'autorité de le stopper selon son gré. C'est grâce à la *Distributed Ledger Technology*, dont nous discuterons le fonctionnement dans un prochain chapitre, que cela est possible. Elle a permis de créer un réseau décentralisé, indépendant, où plusieurs entités peuvent travailler ensemble en toute confiance (TOBIN et REED, 2016).

11. Lorsque des objets sont connectés à Internet, cela s'appelle *"Internet of Things (IOT)"* traduit littéralement en français cela signifie *"Internet des choses"*.

Dans l'article ALLEN, 2016, l'auteur nous propose dix principes de la *SSI* pour tenter d'assurer que le contrôle de l'utilisateur reste au centre de celle-ci.

1. **Existence**. L'utilisateur doit exister indépendamment de la *SSI*. Elle nécessite une identité réelle et il est impossible de digitaliser cette identité dans sa totalité. Seuls certains aspects de celle-ci peuvent être rendus accessibles par la *SSI*.
2. **Contrôle**. L'utilisateur a l'autorité suprême et le contrôle complet sur son identité, il peut choisir ce qu'il veut montrer ou cacher.
3. **Accès**. L'utilisateur doit pouvoir accéder à ses données. Il sait exactement où ses données sont utilisées, dans quel but, et il peut à tout moment demander de les récupérer.
4. **Transparence**. Les systèmes et algorithmes doivent être transparents. Les systèmes doivent être transparent sur leur fonctionnement. Les algorithmes doivent être gratuits, ***open source***, connus ainsi que le plus indépendant possible d'autres architectures.
5. **Persistance**. L'identité doit continuer d'exister tant que l'utilisateur le souhaite.
6. **Portabilité**. Les informations, ainsi que les services liés à l'identité, doivent être transportables. Ceux-ci ne doivent donc pas être détenus par une seule entité tierce.
7. **Interopérabilité**. L'identité ne doit pas être liée à une seule entité. Celle-ci doit pouvoir être utilisée partout où l'utilisateur le souhaite, sans avoir besoin d'en créer une nouvelle.
8. **Consentement**. L'utilisateur doit pouvoir accepter ou refuser chaque utilisation faite de son identité.
9. **Minimisation**. Les informations partagées doivent être minimisées. Si par exemple, une entité souhaite savoir si l'utilisateur a plus de 18 ans, la date de naissance ou l'âge exact ne devraient pas être révélés mais seulement un *oui* ou *non*.
10. **Protection**. Le droit de l'utilisateur doit être protégé. S'il y a un conflit entre les besoins d'un réseau ou ceux d'un utilisateur, le réseau doit préserver les libertés et les droits des individus par rapport aux siens.

Bien évidemment, ce ne sont que des principes. Dans un monde idéal, les solutions de *SSI* les adopteraient tous. Mais comme nous le verrons dans les sections relatives aux quelques solutions disponibles, l'application technique de ces principes peut se révéler pour l'instant compliquée ou incomplète. Dans tous les cas, ceux-ci doivent servir de but idéal à tout système voulant implémenter un système stable et fiable de *SSI*.

L'article TOBIN et REED, 2016, propose également le tableau 2.1 pour la classification de ces dix principes.

Sécurité	**Contrôlabilité**	**Portabilité**
l'identité doit être sécurisée	l'utilisateur contrôle qui voit et accède à ses données	l'utilisateur doit pouvoir utiliser ses données où il le souhaite
Protection	Existence	Interopérabilité
Persistance	Persistance	Transparence
Minimisation	Contrôle	Accès
	Consentement	

TABLE 2.1 – *Tableau de classification des dix principes de la SSI*
Source: *traduit et modifié par l'auteur depuis* TOBIN *et* REED*, 2016*

Ce tableau regroupe les dix principes en trois lignes directrices plus générales que sont la sécurité, la contrôlabilité ainsi que la portabilité. Ce qui peut donner un objectif plus flexible aux premières implémentations d'un système *SSI*.

La *SSI* est donc une nouvelle évolution du modèle d'identité sur Internet, qui est orienté utilisateur et qui promet une nouvelle approche de l'identité numérique avec un point d'honneur sur la protection des données.

2.1.4 Quelques cas d'utilisation

Nous venons de découvrir les bases de ce qu'est la *SSI*. Les domaines d'utilisation de celle-ci sont variés et les cas où elle peut se révéler utile plus que nombreux. Avant de passer à une démonstration d'un cas d'utilisation de la *SSI*, nous allons faire un rapide tour de quelques impacts qu'elle pourrait avoir dans différents domaines.

Banque et finance

Dans le secteur bancaire et financier, l'implémentation de la *SSI* peut ouvrir de nouvelles possibilités, aux individus comme aux institutions.

L'accès simplifié à divers marchés est alors possible. Nous pouvons donc souscrire à différents services financiers ou bancaires, comme *Visa et Mastercard*[12], en partageant une identité commune sur ceux-ci. Cela évite également d'avoir à s'inscrire indépendamment sur chaque institution pour pouvoir les utiliser. C'est donc à la fois un gain de temps pour le client ainsi qu'un gain de nouveaux clients potentiels pour l'institution (PREUKSCHAT et REED, 2021).

En ce qui concerne les cartes de crédit, il devient possible pour l'institution d'automatiser la vérification des informations de leur demandeur. Cela permet un processus métier accéléré et une meilleure gestion des fraudes à la carte de crédit. Cette amélioration est possible car

12. Ce sont des sociétés de crédits qui offrent des cartes de débit et crédit.

l'institution financière peut choisir d'accepter des requêtes de clients dont les informations proviennent d'émetteurs à qui elle fait confiance. Notons que cet exemple peut aussi s'appliquer aux demandeurs de prêt bancaire. Les processus ***Know your customer*** et ***Anti-Money Laudering*** en sont directement impactés (PREUKSCHAT et REED, 2021).

De leur côté, les systèmes de paiement sont eux-aussi impactés. Le paiement en ***peer-to-peer*** peut alors être combiné à la vérification d'information grâce à la *SSI*. Ce qui permet, par exemple, de trouver et de payer un plombier de confiance rapidement (WILSER, 2020).

E-commerce

D'année en année, le commerce en ligne continue son expansion. Le téléphone portable est devenu un outil indispensable sur les sites d'e-commerce [13]. C'est à travers lui que la majorité des transactions sont effectuées (MARFICE, 2020).

Avec la *SSI*, les sites peuvent alors offrir une expérience personnalisée à chaque client, si ceux-ci acceptent de partager des informations sur leur identité. L'expérience de l'utilisateur en est alors améliorée et ses données sont respectées (WILSER, 2020).

Cette expérience est aussi améliorée en permettant à l'utilisateur de se connecter, grâce à une identité unique et sans mot de passe, sur les différents sites de commerce. Notons cependant que cela n'est pas spécifique à la *SSI* et que les solutions telles que *FIDO*, dont nous discutons dans le chapitre 2.3.3, et *OpenID Connect* le permettent également. L'utilisateur et le commerce savent également s'ils peuvent se faire confiance mutuellement en vérifiant leur identité numérique, ce qui permettrait d'éviter les arnaques en ligne (PREUKSCHAT et REED, 2021).

Réseaux Sociaux

L'expérience utilisateur sur les réseaux sociaux va également se voir bouleverser avec l'arrivée de la *SSI*. Plus besoin de s'inquiéter des conséquences du partage d'informations sur Internet. L'utilisateur pourra choisir les données qu'il souhaite diffuser et savoir pourquoi celles-ci sont utilisées et par qui (WILSER, 2020).

L'introduction de nouveaux concepts sociaux, comme celui du monde virtuel [14], font également écho à la nécessité de technologies comme la *SSI* (BEDOYA, 2022). Ainsi, le manque d'une couche d'identité sur Internet semble se répercuter sur toutes les technologies se basant dessus.

13. Ce sont des commerces en ligne.
14. Les métavers sont des mondes virtuels où les utilisateurs peuvent interagir, avec d'autres entités ou entre eux, dans un espace modélisé sur Internet.

Santé

Dans le monde de la santé, les questions de protection et transmission de données médicales, sur Internet, se posent comme un réel défi.

Le portefeuille numérique, qu'amène la *SSI*, permet un stockage et partage sécurisé des informations médicales. Il devient, par exemple, possible d'obtenir un certificat médical directement sur son portefeuille numérique après une consultation chez son médecin (PREUKSCHAT et REED, 2021).

Le patient peut également stocker et partager ses documents médicaux en quelques pressions du doigt. Lorsqu'il veut les transmettre, il peut choisir ce qui est envoyé et vérifier l'identité de celui à qui ces informations sont partagées (PREUKSCHAT et REED, 2021).

La *SSI* peut également réduire les coûts des opérations médicales en supprimant, par exemple, des étapes inefficaces, comme l'enregistrement d'un patient, lors du processus d'accueil des patients (SHUAIB, ALAM, SHABBIR ALAM et SHAHNAWAZ NASIR, 2021).

Tourisme et voyage

Même si l'épidémie récente a momentanément réduit les voyages internationaux, il n'en est pas moins certain que le tourisme de masse est une réalité du monde actuelle. Ce nombre de voyageurs croissants représente un défi technique pour toutes les infrastructures liées au tourisme, comme les organismes de transports ou les hôtels.

La *SSI* se présente également comme l'une des solutions permettant de simplifier ces défis techniques. Elle peut, par exemple, accélérer tous les processus d'enregistrement lors d'un voyage en les automatisant. Ainsi, s'enregistrer à l'aéroport, réserver une chambre d'hôtel, louer une voiture ou même acheter un billet pour un concert, deviennent des processus plus accessibles. Il suffit de sortir son téléphone portable pour effectuer ces opérations et avoir la garantie de retrouver rapidement ces enregistrements, stockés alors dans notre portefeuille numérique de manière sécurisée (PREUKSCHAT et REED, 2021). Notons cependant que certaines compagnies aériennes permettent d'effectuer ces différentes opérations en avance, lors d'une phase de pré-enregistrement. La *SSI* offre alors, dans ce cas, uniquement une nouvelle expérience d'enregistrement à l'utilisateur (AIR FRANCE, s. d.).

Les organisations étatiques sont-elles aussi soulagées. Le traitement des renouvellements des passeports ou des cartes d'identité sont rapidement traités. La *SSI* permet alors de se connecter à l'organe émettant ces documents pour obtenir une mise à jour de ceux-ci. C'en est alors terminé des longues files d'attentes au guichet de ces institutions.

2.1.5 Démonstration

Dans cette section, nous effectuons une démonstration d'un cas d'utilisation de la *SSI*. Les captures qui suivent ont été prise par l'auteur en effectuant la démonstration disponible publiquement sur Animo Demo.

Avant-propos

Nous ne prétendons en aucun être les créateurs de cette démonstration, nous l'exposons dans ce travail pour permettre une meilleure compréhension du fonctionnement de la *SSI*. Tous les droits liés à la démonstration vont à la société *Animo Solutions* dont les informations supplémentaires sont disponibles sur Animo.

Mise en place

La première étape avant de pouvoir lancer la démonstration sur le site est d'installer un portefeuille numérique sur son téléphone portable. *Animo Solution* nous propose ici deux choix de portefeuilles.

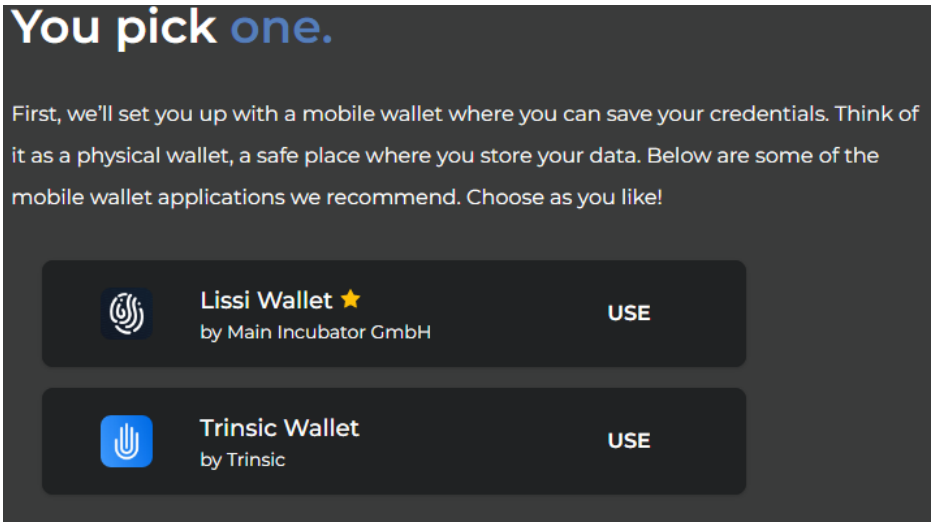


FIGURE 2.7 – *Choix de portefeuille Animo*
Source: *de l'auteur*

Le portefeuille utilisé pour la démonstration de ce travail est le *Lissi Wallet*. La figure 2.8 représente l'écran d'accueil de l'application.

FIGURE 2.8 – *Écran d'accueil du portefeuille numérique Lissi*
Source: *de l'auteur*

Une fois le portefeuille choisi, nous pouvons passer à l'étape suivante qui consiste à choisir un personnage pour la démonstration. Dans ce travail, nous avons choisi d'être *Joyce*.

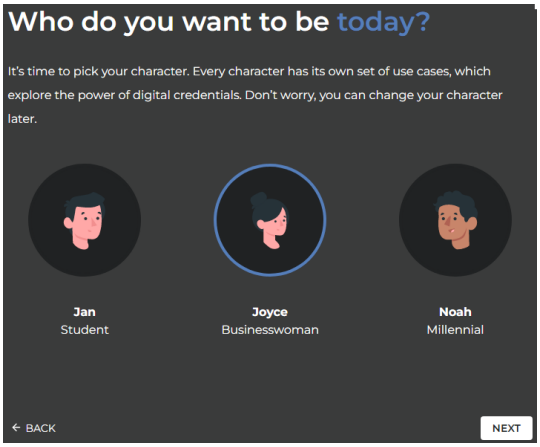


FIGURE 2.9 – *Choix de personnage Animo*
Source: *de l'auteur*

Quand le personnage est choisi, un code QR nous est présenté. Il faut utiliser le scanneur de code QR disponible sur le portefeuille numérique pour le lire.

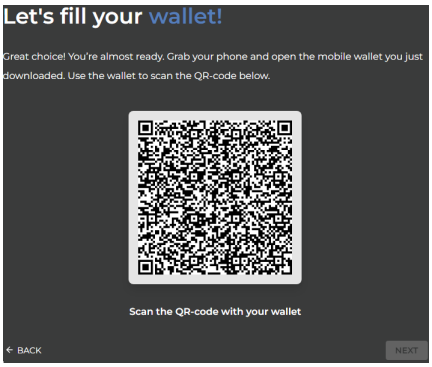


FIGURE 2.10 – *Code QR de connexion Animo*
Source: *de l'auteur*

Une fois ce code scanné, l'application nous demande alors d'accepter la connexion.

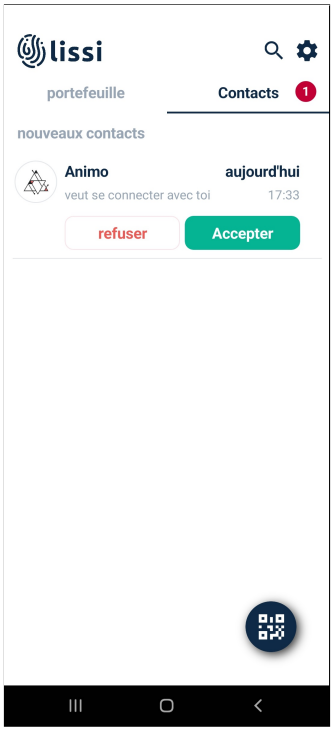


FIGURE 2.11 – *Demande de connexion Lissi*
Source: *de l'auteur*

Après que la connexion soit acceptée, des identifiants nous sont transmis.

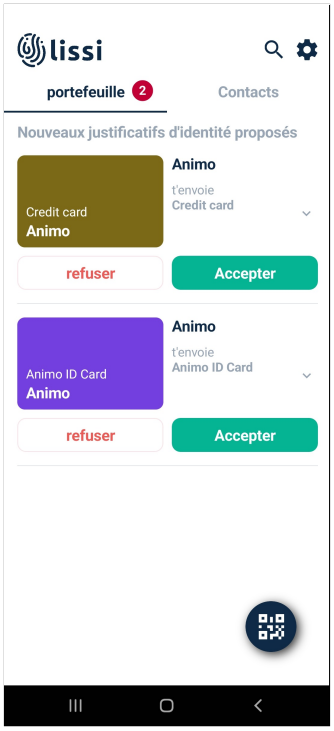


FIGURE 2.12 – *Proposition d'identifiants Lissi*
Source: *de l'auteur*

Nous pouvons maintenant choisir un scénario de démonstration. Dans ce travail, nous avons choisi la réservation d'une chambre d'hôtel.

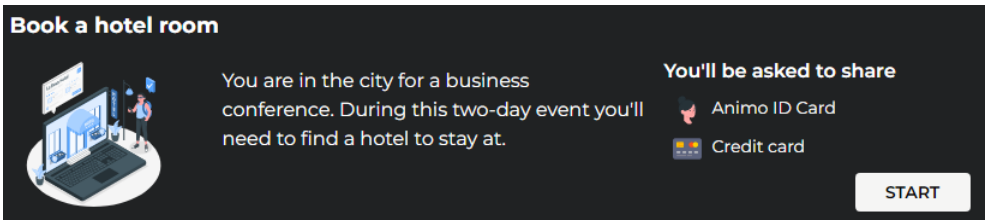


FIGURE 2.13 – *Choix de scénario Animo*
Source: *de l'auteur*

Réservation d'une chambre d'hôtel

L'étape suivante consiste à créer une connexion avec l'hôtel.

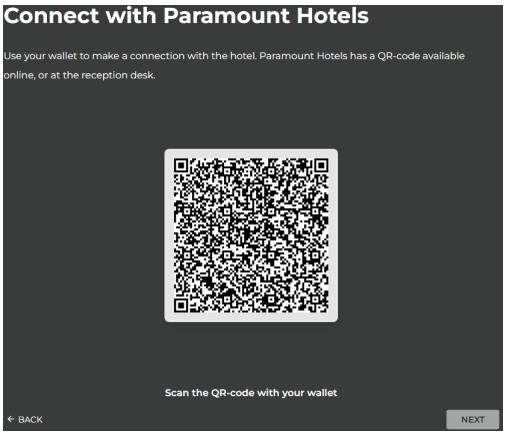


FIGURE 2.14 – *Code QR de connexion à l'hôtel Animo*
Source: *de l'auteur*

L'application nous demande alors d'accepter la connexion avec l'hôtel.

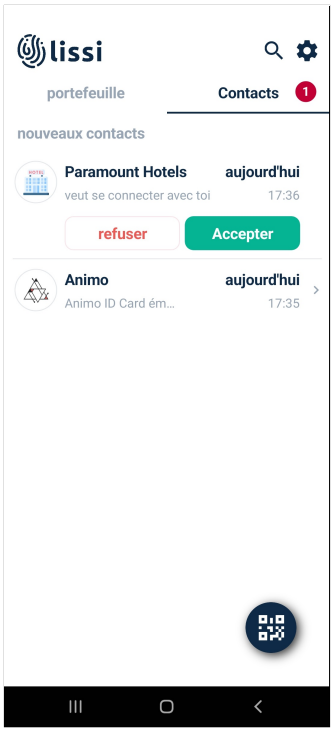


FIGURE 2.15 – *Demande de connexion de l'hôtel Lissi*
Source: *de l'auteur*

Pour réserver une chambre, l'hôtel va nous demander que nous lui transmettions des informations, comme notre carte de crédit.

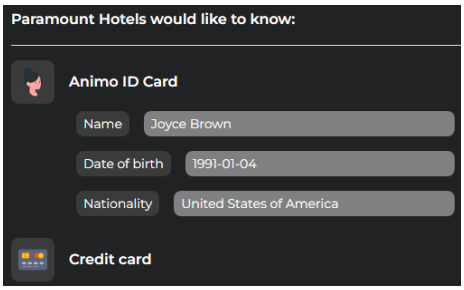


FIGURE 2.16 – *Demande d'identifiants Animo*
Source: *de l'auteur*

Nous pouvons voir le détail des informations demandées ainsi qu'accepter ou refuser leur envoi sur l'application.

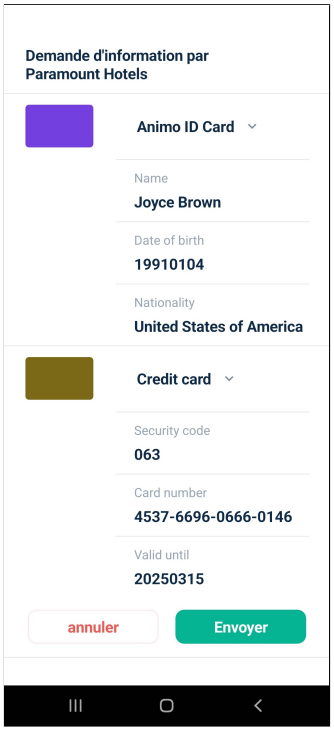


FIGURE 2.17 – *Demande d'identifiants Lissi*
Source: *de l'auteur*

Si les informations envoyées à l'hôtel sont acceptées, nous recevons alors une carte pour ouvrir la chambre.

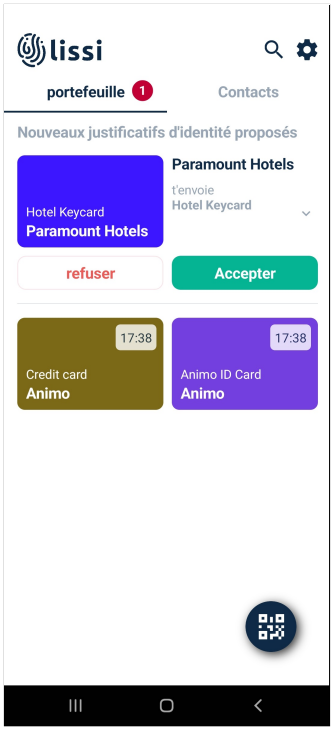


FIGURE 2.18 – *Proposition d'une keycard pour la chambre Lissi*
Source: *de l'auteur*

La chambre est désormais réservée et tous les documents nécessaires se trouvent dans notre portefeuille électronique.

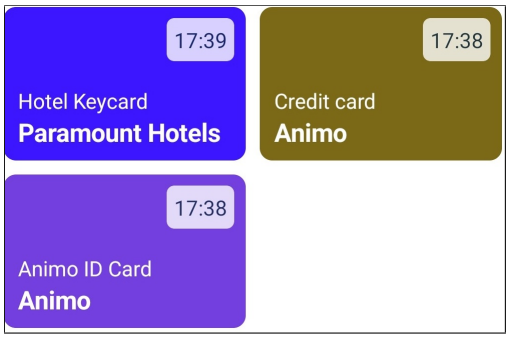


FIGURE 2.19 – *Documents possédés Lissi*
Source: *de l'auteur*

À prendre avec soi

Cette démonstration donne une idée de la puissance de la *SSI*. Nous avons donc vu comment nous pouvions rapidement réserver une chambre d'hôtel grâce à un portefeuille numérique. La plupart des scénarios se jouent alors selon les étapes suivantes :

1. scanner le QR code d'invitation ;
2. accepter ou refuser la connexion ;
3. accepter ou refuser l'envoi d'informations ;
4. recevoir l'élément désiré.

Nous allons voir dans la prochaine section comment cela fonctionne en arrière-plan.

2.1.6 Fonctionnement

Nous tentons dans cette section de comprendre le fonctionnement de la *SSI*. Pour ce faire, il est nécessaire d'établir progressivement les différents blocs qui la composent. La compréhension de ceux-ci permet de mieux concevoir les relations utilisées dans la *SSI*.

Comme nous l'avons dit à quelques reprises au cours de ce travail, certaines des technologies utilisées sont en place depuis des années déjà. Cependant, la combinaison de celles-ci pour former le modèle de l'identité autogérée est une idée récente (PREUKSCHAT et REED, 2021).

Selon le livre PREUKSCHAT et REED, 2021, il y a sept blocs qui forment la *SSI*.

1. Les justificatifs vérifiables appelés *Verifiable Credentials (VC)* .
2. Le triangle de confiance *(trust triangle)* de la *figure 2.22*, entre les émetteurs *(issuers)*, détenteurs *(holders)* et vérificateurs *(verifiers)*.
3. Le portefeuille numérique appelé *digital wallet*.
4. L'agent numérique appelé *digital agent*.
5. L'identifiant décentralisé appelé *Decentralized Identifier (DID)*.
6. La technologie de registre distribué, appelée Distributed Ledger Technology (DLT), que nous traiterons dans une prochaine section car son impact sur la *SSI* mérite une analyse particulière.
7. Les règles de gouvernance appelées *governance frameworks*.

Pour le reste de ce document, nous allons utiliser la dénomination anglaise de ces blocs. Notons que nous n'entrerons pas dans les détails techniques complexes du fonctionnement de ceux-ci. En effet, ce n'est tout d'abord pas le but de ce document, et ensuite parce que le traitement complet de chacun de ces blocs mériterait un travail de recherche à part entière.

Des justificatifs comme dans la vraie vie

Un *Verifiable Credentials* est l'équivalent numérique d'un justificatif dans le monde physique, mais qu'est-ce qu'un justificatif exactement ?

Nous en possédons tous, que ce soit une carte de crédit, un passeport ou même une carte d'abonnement de salle de sports. Ces justificatifs offrent plusieurs types d'informations différentes (W3C, 2022b).

1. Des informations sur le ***sujet*** du justificatif, par exemple un nom, un numéro d'étudiant, etc...
2. Des informations sur l'autorité émettrice du justificatif, sur le passeport par exemple nous pourrions avoir le nom de l'agence nationale.
3. Des informations sur le type de justificatif, par exemple si c'est un passeport, une carte de cinéma ou un permis de conduire.
4. Des informations sur les contraintes du justificatif, par exemple la date d'expiration du passeport.

Ainsi, plus généralement, un justificatif peut être défini comme un ensemble d'informations [15] qu'une autorité revendique comme vrai sur un ***sujet*** [16] et qui lui permet ensuite de prouver aux autres entités que ses informations sont vraies. Il est également commun de trouver sur le justificatif une signature qui prouve son authenticité et celle de l'autorité qui l'a émise, par exemple un hologramme sur les passeports (PREUKSCHAT et REED, 2021). La figure 2.20 donne un exemple de justificatif.

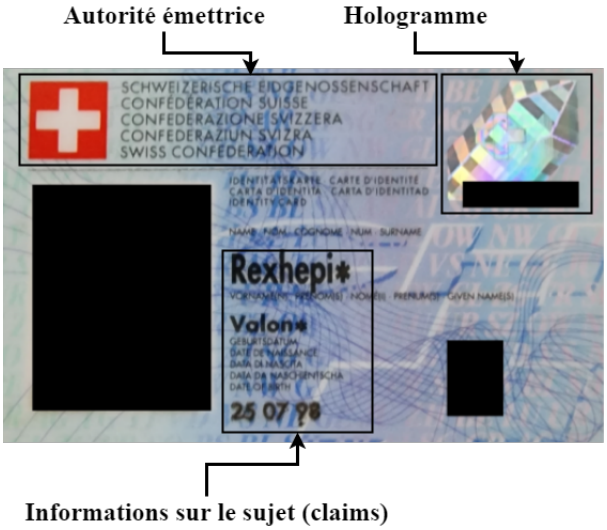


FIGURE 2.20 – *Exemple d'un justificatif (carte d'identité)*
Source: *de l'auteur*

15. En anglais, les informations sont appelées *"claims"*.
16. Notons que le sujet peut être une personne, un animal, une entreprise ou n'importe quel type d'entité sur laquelle il peut y avoir des informations.

Un *VC* a la même fonction. Il sert à représenter un justificatif que nous pourrions avoir dans le monde physique mais dans le monde numérique. Bien évidemment, le fait que celui-ci soit numérique permet également de rajouter des couches technologiques qui le rendent ainsi plus fiable que sa version physique (W3C, 2022b).

Les standards définis dans W3C, 2022b indiquent que les composants de base du *Verifiable Credentials* sont les suivants :

1. les ***métadonnées*** du justificatif, qui vont donner des informations sur les propriétés du justificatif, par exemple la date d'expiration ;
2. les ***claims***, qui sont les informations sur le ***sujet*** ;
3. les ***proofs***, qui sont les preuves de la validité des *claims*.

Le code suivant représente un exemple de la structure d'un *VC*[17].

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "id": "CredentialId",
  "type": "CredentialType",
  "issuer": "CredentialIssuer",
  "issuanceDate": "CredentialDate",
  "credentialSubject": {
    "id": "SubjectId"
  },
  "proof": {
    "type": "ProofType",
    "created": "ProofCreatedDate",
    "proofPurpose": "ProofPurpose",
    "verificationMethod": "ProofVerificationMethod"
  }
}
```

La structure utilisée ici est bien évidemment incomplète[18], mais elle nous permet déjà de constater que la structure numérique d'un justificatif a pour but d'imiter la version physique de celui-ci. Dans l'exemple, nous pouvons voir que nous avons un *id* qui est l'identifiant de justificatif ou encore une ***proof*** qui sert à prouver sa validité. Nous n'allons pas plus rentrer dans le détail sur la structure des *VC* dans ce document.

Avant de passer au bloc suivant, il est important que nous fassions une pause sur les ***proofs***. Une ***proof*** permet de démontrer qu'une information est vraie en utilisant des mécanismes cryptographiques (PREUKSCHAT et REED, 2021).

17. La structure est en *JSON for Linking Data (JSON-LD)*.
18. Les exemples sur W3C, 2022b sont plus complets et chaque propriété possible de la structure est également détaillée.

Un type de ***proof*** cryptographique, qui est étudié et envisagé pour la *SSI*, est celui du *Zero-Knowledge Proofs (ZKP)*[19]. Ce *ZKP* permet à une entité de prouver mathématiquement à une autre entité la véracité d'une information, sans en révéler plus que la véracité de l'information en question. Nous allons utilisé une métaphore pour tenter de comprendre ce que cela signifie. Celle-ci est inspirée de l'exemple d'Oded Goldreich dans l'article GOLDREICH, 2003.

Imaginons deux objets quasiment identiques mais dont la couleur diffère, l'un est vert et l'autre est rouge. Nous voulons prouver à notre ami daltonien que ces deux objets sont différents, sans jamais lui révéler la couleur de l'objet. Nous allons donc demander à notre ami de prendre les objets, les cacher et tour après tour, nous montrer l'un des deux objets. À chaque tour, nous allons lui dire si l'objet montré est différent de celui montré précédemment en disant seulement "c'est le même" ou "il est différent". Plus le nombre de tours grandira, plus il sera difficile pour nous de ne jamais nous tromper si ceux-ci sont évidemment les mêmes objets avec la même couleur. La couleur étant différente, nous n'allons jamais nous tromper, prouvant ainsi à notre ami daltonien que les objets ne sont pas identiques sans jamais donner d'autres informations que "c'est le même" ou "il est différent".

Ainsi, nous comprenons rapidement l'utilité du *ZKP* avec la *SSI*. En effet, le contrôle et la protection des données de l'utilisateur sont, par exemple, renforcés avec ce type de mécanisme. De plus, il y a quatre notions importantes, utilisées par la *ZKP*, que nous devons découvrir avant de passer à la suite (PREUKSCHAT et REED, 2021).

1. La ***selective disclosure***[20] permet de ne révéler que certains ***claims*** lors d'un échange. Par exemple, lorsque la salle de sports nous demande nos informations, nous souhaitons ne révéler que notre date de naissance.
2. La ***predicate proof***[21] est une preuve qui répond à une question fermée. Par exemple, si la salle de sports veut savoir si nous sommes majeurs, nous n'allons pas transmettre notre date de naissance mais simplement répondre par "oui" ou "non" à la question "Êtes-vous majeur ?".
3. La ***révocation*** permet de révoquer un justificatif. Par exemple, notre permis, si celui-ci a été retiré, ne sera plus valide lorsqu'il sera contrôlé.
4. L'***anti-corrélation*** empêche de faire de la corrélation avec les données de l'utilisateur en minimisant les informations exposées. Par exemple, l'abonnement à un journal et la tendance de vote d'une personne.

Le *VC* est donc un élément complexe qu'il soit physique ou numérique. Il doit également être sécurisé et unique pour que, comme dans le monde physique, celui-ci ne puisse pas être reproduit, volé ou utilisé par une autre personne.

19. En français, c'est la *"preuve à divulgation nulle de connaissance"*.
20. En français, c'est la *"divulgation sélective"*.
21. En français, c'est la *"preuve par prédicat"*.

Une confiance au centre de l'identité autogérée

Comme dans le monde physique, le monde numérique nécessite une confiance entre les différents acteurs pour fonctionner. L'écosystème de la *SSI* est basé sur la relation entre trois entités : les émetteurs (issuers), détenteurs (holders) et vérificateurs (verifiers) (W3C, 2022b).

Les émetteurs, en anglais *issuers*, sont les entités qui vont être la source d'un justificatif. Pour un passeport, l'émetteur est une agence nationale par exemple. Dans l'écosystème de la *SSI*, les individus (de même que les choses) peuvent également être des *issuers*, au même titre qu'un serveur peut délivrer un certificat (PREUKSCHAT et REED, 2021). Ils vont émettre des ***claims*** sur le ***sujet***, créer un *VC* à partir de ces informations et transmettre ce justificatif à un détenteur (W3C, 2022b).

Les détenteurs, en anglais *holders*, sont les entités qui vont détenir des *VCs*. Ils sont, dans la plupart des cas, également le sujet du *VC* détenu. Par exemple, si nous avons un bébé et que celui-ci à un *VC*, nous allons être le détenteur mais le bébé sera le ***sujet*** (PREUKSCHAT et REED, 2021). Les détenteurs stockent les justificatifs dans un dépôt prévu à cet effet, par exemple un portefeuille numérique (W3C, 2022b).

Les vérificateurs, en anglais *verifiers*, sont les entités à qui les *VCs* vont être présentés. Ce processus se fait à travers des présentations qui sont des conteneurs d'un ou plusieurs *VCs* envoyés à un *verifier* (W3C, 2022b). La figure 2.21 représente l'échange entre ces différentes entités dans l'écosystème de la *SSI*.

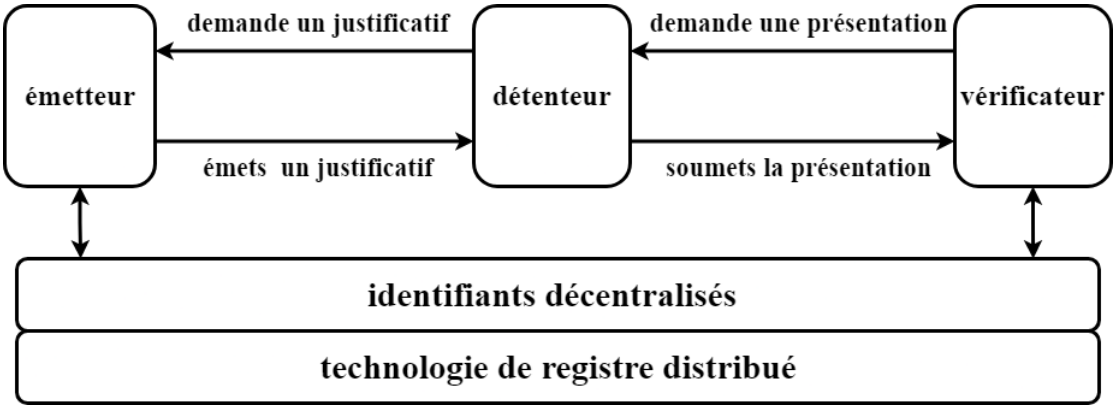


FIGURE 2.21 – *Échanges entre les différentes entités de l'écosystème de la SSI*
Source: *de l'auteur inspirée par la figure 2.3 de* PREUKSCHAT *et* REED*, 2021*

Concentrons-nous sur la partie supérieure de ce schéma, celle-ci représente la relation entre les trois entités pour l'échange de *VCs*. Ainsi, nous voyons que l'émetteur reçoit une demande de justificatif et l'émet. Le détenteur fait la demande de justificatif et le reçoit par l'émetteur. Il s'occupe également de l'envoie d'une présentation lorsque celle-ci est demandée. Finalement, le vérificateur demande une présentation et la reçoit. Bien évidemment, les entités peuvent briser ce schéma à n'importe quel moment. Un émetteur peut donc tout à fait refuser d'émettre un justificatif, tout comme un détenteur peut refuser de soumettre une présentation.

Il est cependant important de noter que l'opération réussit uniquement si le *verifier* fait confiance à l'*issuer*, comme dans le monde physique d'ailleurs. Si nous sommes arrêtés par les forces de police et que nous présentons un permis de conduire qui n'est pas délivré par l'organisme compétent, cela risque de mal se passer. Cette relation est appelée *trust triangle*, le triangle de confiance (PREUKSCHAT et REED, 2021). Celui-ci peut être représenté par la figure 2.22.

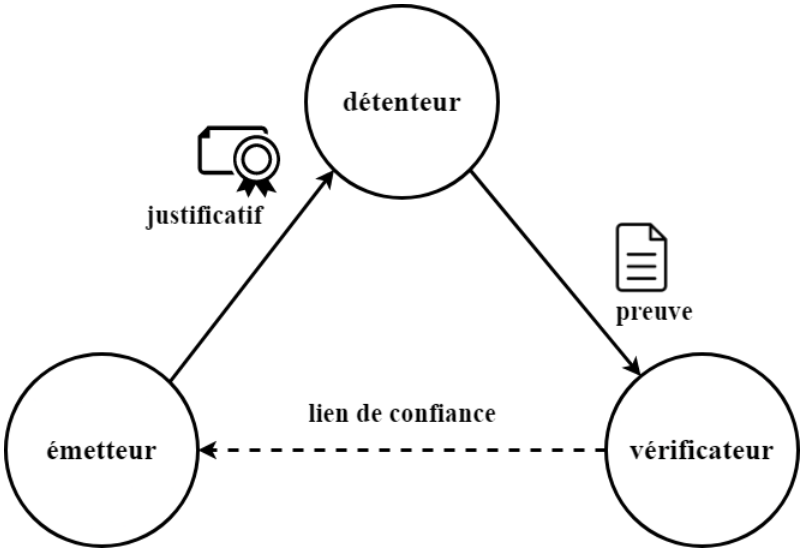


FIGURE 2.22 – *Triangle de confiance*
Source: *de l'auteur inspirée par la figure 2.4 de* PREUKSCHAT *et* REED*, 2021*

Ainsi, nous voyons dans ce schéma que le lien de confiance entre le vérificateur et l'émetteur est primordial. Sinon les preuves reçues par le vérificateur du détenteur ne sont pas considérées comme valables, brisant de ce fait le triangle de confiance.

Ajoutons également que durant un échange, par exemple lorsque nous voulons réserver une chambre d'hôtel sur Internet, nous allons être, en tant que client, à la fois détenteur et vérificateur. *Holder* parce que nous présentons des justificatifs à l'hôtel (identité, carte de crédit, etc...) et *verifier* car l'hôtel va nous présenter ses justificatifs pour prouver que le site internet n'est pas un faux (PREUKSCHAT et REED, 2021).

Finalement, en voyant ces schémas, nous pourrions croire à un modèle fédéré, mais ce n'est pas le cas. En effet, il est important de noter que lors d'un échange, seul le *holder* et le *verifier* se transmettent des informations, l'*issuer* ne joue donc pas de rôle intermédiaire entre les deux. Cela signifie qu'il n'a aucune connaissance de la communication en cours entre les deux autres acteurs.

Un portefeuille... mais numérique

Dans le monde physique, nous rangeons, pour la plupart, nos différents documents dans un portefeuille. Que ce soit la carte de la salle de sports ou les cartes bancaires, nous voulons qu'elles soient accessibles rapidement et qu'elles soient toujours en sécurité au plus proche de nous. Les mêmes idées s'appliquent dans le monde numérique (PREUKSCHAT et REED, 2021).

L'une des premières solutions trouvées, pour implémenter et proposer un portefeuille numérique, a vu le jour dans un autre objet que nous avons toujours près de nous, le téléphone portable. Ce que nous appelons en anglais les *mobile wallets*, en français les *portefeuilles mobiles*, s'imposent de plus en plus comme les leaders dans le marché de portefeuilles digitaux. Pourquoi ? Simplement, parce que ceux-ci sont directement inclus dans votre téléphone portable. Pour les téléphones de la marque *Apple*, c'est le *Apple Wallet* et pour les téléphones *Android*, c'est le *Google Pay* (PREUKSCHAT et REED, 2021).

FIGURE 2.23 – *Application Apple Wallet*
Source: *de l'auteur*

Ces marques ont bien compris l'enjeu que représentent ces applications, *Apple* à même récemment mis à jour son *mobile wallet*. Il propose désormais, en plus de pouvoir payer avec, de stocker nos identifiants et même d'échanger une clé de chambre d'hôtel en un simple message (APPLE, 2022b). Il existe également d'autres applications qui ont le même but, comme le *Lissi Wallet* que nous avons vu dans la démonstration.

L'implémentation de ces *wallets* soulève des questions sur les avatanges et inconvénients liés à celle-ci. Certaines applications utilisent un stockage interne au téléphone pour mieux sécuriser les informations contenues dans le *wallet*, c'est ce que nous appelons des *edge wallet*. Cependant, cette technique implique qu'il faut trouver des solutions pour synchroniser le contenu du portefeuille entre différents appareils. Une autre approche est de stocker le *wallet* sur Internet et d'y accéder via des requêtes qui peuvent être faites depuis n'importe quel type de dispositif connecté, ce sont des *cloud wallet*. Cette technique, même si elle règle le problème de la portabilité, peut amener des problèmes de sécurité et de confidentialité (W3C, 2022a).

La sécurisation de ces *wallets* représente alors un défi technique pour les développeurs. Les applications mobiles peuvent utiliser un panel de technologies disponible sur les dernières générations de téléphones portables pour les sécuriser. Ces technologies vont du simple code à quatre chiffres jusqu'à la reconnaissance biométrique.

Le livre PREUKSCHAT et REED, 2021 propose les lignes directrices suivantes sur la façon de penser le *mobile wallet* :

1. un portefeuille numérique doit être capable d'accepter n'importe quel type de *VC* ;
2. un portefeuille numérique doit pouvoir être installé et utilisé sur tous les types d'appareils que nous souhaitons utiliser et le contenu de celui-ci doit être synchronisé entre tous les dispositifs ;
3. le contenu du portefeuille numérique doit pouvoir être sauvegardé et exporté sur un autre portefeuille numérique, si nous décidons par exemple de passer de l'*Apple Wallet* vers *Google Pay* ;
4. l'expérience utilisateur doit être la même peu importe le *wallet* utilisé, et ceux-ci doivent tous proposer les mêmes fonctionnalités.

Cependant, il y a une différence fondamentale entre un portefeuille numérique et physique, et cette différence demeure dans la façon de le contrôler. Nous allons voir cela dans la prochaine section.

Un agent numérique

Un portefeuille physique est contrôlé par son propriétaire. Lorsque nous souhaitons présenter notre carte d'identité, c'est en général nous qui sortons le portefeuille et la carte. Pour un *wallet* numérique, c'est différent.

Les machines ne comprennent que des langages spécifiquement conçus pour elles, et les êtres humains ne parlent pas ces langages. Il est donc techniquement très complexe de contrôler directement son portefeuille numérique. C'est à travers ce que nous appelons un *digital agent* que nous allons contrôler le *wallet*. Cet agent est un programme qui va traduire nos instructions pour le *digital wallet*. Il va permettre également de sécuriser le portefeuille en s'assurant que l'entité qui souhaite l'utiliser est bien son propriétaire (PREUKSCHAT et REED, 2021).

Le schéma 2.24 représente la relation entre une personne, un agent et un *wallet*.

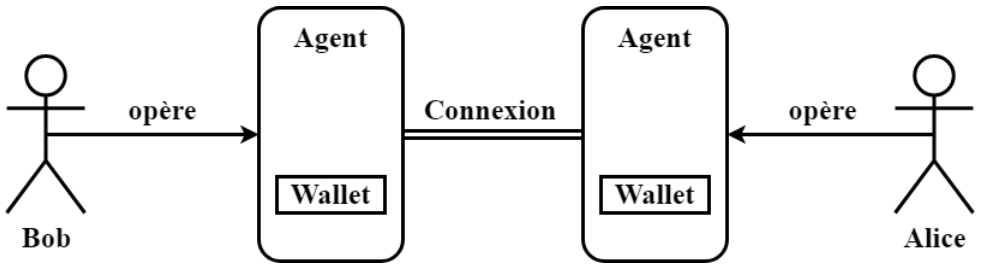


FIGURE 2.24 – *Relation entre une personne, un agent et un wallet*
Source: *de l'auteur inspirée par la figure 2.8 de* PREUKSCHAT *et* REED*, 2021*

Nous pouvons voir dans ce schéma *Bob* et *Alice*. Ils vont donner des instructions à leur agent respectif. Cet agent englobe leur *wallet*, il peut donc le contrôler. Nous voyons également que lors d'un échange entre deux entités, ce ne sont pas les portefeuilles qui se connectent entre eux, mais les agents.

Comme pour les *digital wallet*, il existe plusieurs types d'agent. Le *edge agent* se trouve localement dans l'appareil du *holder*, tandis que le *cloud agent* est stocké sur Internet chez des fournisseurs spécialisés appelés *agence*. Le schéma 2.25 montre les interactions entre les différents types d'agents.

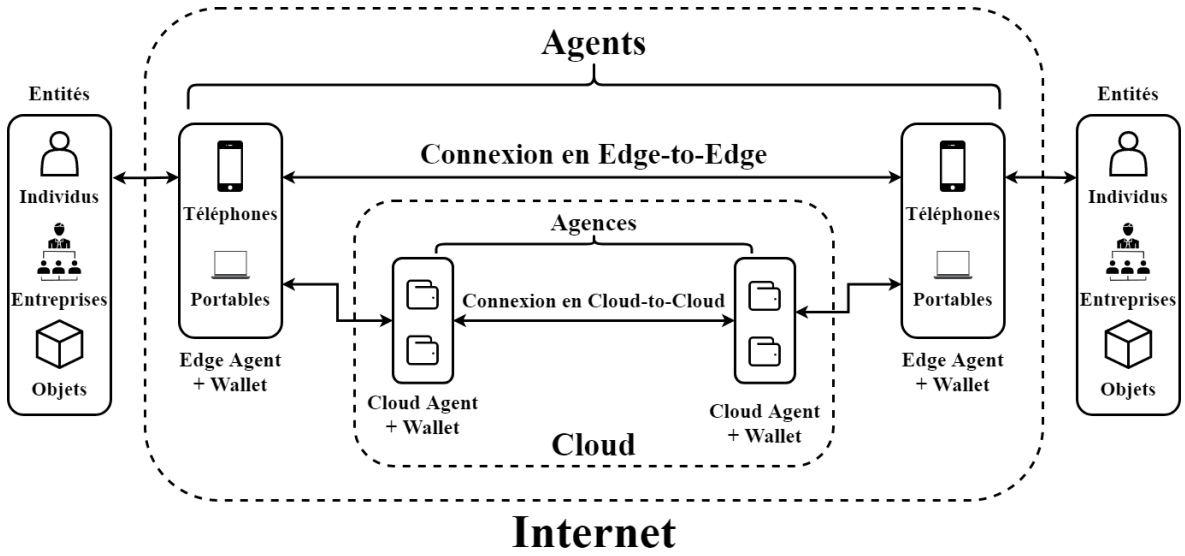


FIGURE 2.25 – *Relations entre les différents types d'agents*
Source: *de l'auteur inspirée par la figure F de* SOVRIN*, 2019*

La compréhension de cette notion d'agent est importante car, nous le verrons plus tard dans les solutions disponibles, cet agent joue un rôle primordial dans la mise en place d'un système pour la *SSI*.

Un identifiant pour les gouverner tous

Lorsque nous avons discuté de la problématique d'Internet, nous avons vu que le système d'identification actuel via les adresses *IP* ne permet pas d'identifier l'entité derrière l'écran. Ainsi, l'utilisation de cette adresse comme identifiant pour la *SSI* est plus que limitée. Il a donc fallu créer un nouveau type d'identifiants qui est permanent, résoluble, vérifiable cryptographiquement et décentralisé. Le *Decentralized Identifier (DID)* est né (PREUKSCHAT et REED, 2021).

Un *DID* peut être utilisé pour représenter n'importe quel ***sujet*** sur Internet. Cela va de la représentation d'une personne, à un document sur un serveur. Ce nouvel identifiant unique permet également aux entités de générer elles-mêmes leurs propres *DIDs*. Bien évidemment, les entités peuvent avoir autant de *DIDs* que nécessaire (W3C, 2021).

La structure du *DID* est également nouvelle. Elle est constituée d'un schéma, un identifiant de méthode ainsi qu'un identifiant unique spécifique à la méthode (W3C, 2021). Le schéma 2.26 donne un exemple de la structure d'un *DID*.

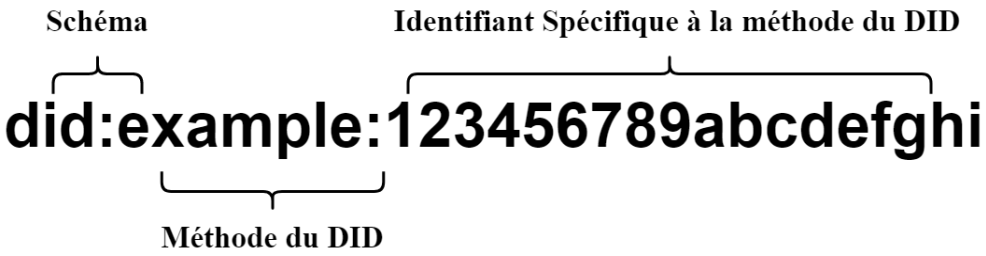


FIGURE 2.26 – *Exemple de structure d'un Decentralized Identifier*
Source: *traduit et modifié par l'auteur depuis* W3C*, 2021*

Une méthode *DID* indique le mécanisme par lequel un type particulier de *DID* ainsi que le document associé sont créés, résolus, mis à jour et désactivés (W3C, 2021). Un *DID*, lorsqu'il est résolu, donne un *document DID* qui a une structure en *JSON-LD*. Voilà un exemple, récupéré depuis W3C, 2021, de document *DID*.

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ]
  "id": "did:example:123456789abcdefghi",
  "authentication": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "Ed25519VerificationKey2020",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyMultibase": "zH3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }]
}
```

Le schéma 2.27 représente les composants de l'architecture du *DID*.

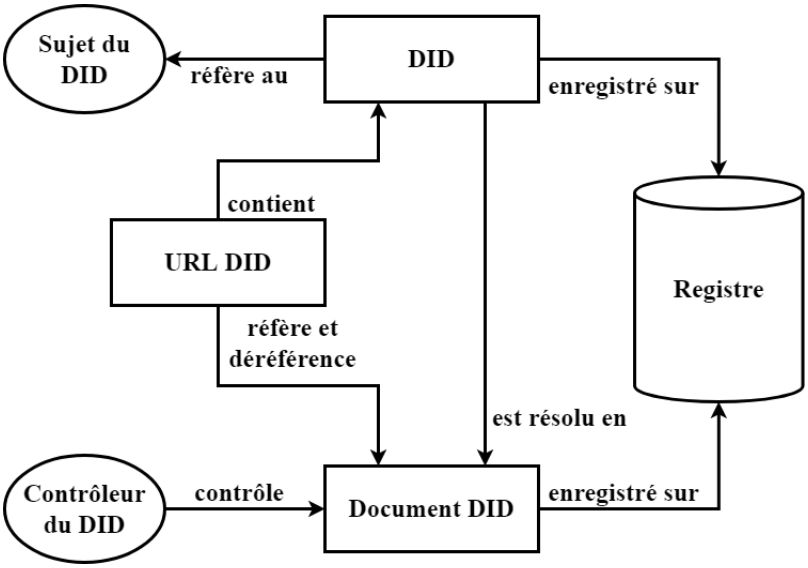


FIGURE 2.27 – *Architecture des composants du Decentralized Identifier*
Source: *traduit et modifié par l'auteur depuis* W3C*, 2021*

Le contrôleur est l'entité qui contrôle le document, en lui apportant des modifications par exemple selon la méthode définie. Ce document ainsi que le *DID* peuvent être écrits dans un registre [22]. Le document peut également être reconstruit, sans être au préalable enregistré, grâce au *DID* et à l'historique des ***smart contract***. Un nouvel élément, dont nous n'avons pas parlé auparavant, est le *DID* ***Uniform Resource Locator (URL)***. Celui-ci étend les capacités d'un *DID* en lui permettant d'inclure dans sa structure des composants de l'*URL*. Finalement, un *DID* réfère à un sujet qui, comme nous l'avons vu, peut être n'importe quel type d'entité. Notons que le sujet et le contrôleur peuvent être la même entité (W3C, 2021).

Avant de passer à la section suivante, il est intéressant de mentionner que la création des *DIDs* a également donné naissance à un type de communication appelé *DIDComm*. Celle-ci, comme le protocole *TCP/IP*, établit une connexion, mais entre des *DIDs*. Le *DIDComm* permet une relation permanente, privée, sécurisée et extensible (PREUKSCHAT et REED, 2021).

22. Par exemple, c'est une *Distributed Ledger Technology*.

Un cadre et des règles pour une confiance maximale

Le dernier bloc utilisé dans la *SSI* dont nous parlerons dans cette section est celui des règles de gouvernance, en anglais *governance frameworks*. Ce sont des règles légales, techniques ou commerciales qui permettent de renforcer la confiance entre les entités. C'est par exemple les procédures à appliquer pour émettre un nouveau justificatif ou bien les conditions qu'une entité doit respecter pour en posséder un. L'entité qui va établir ces règles est appelée *autorité de gouvernance* ou en anglais *governance authority* (PREUKSCHAT et REED, 2021).

Ainsi la figure 2.28 représente l'impact de la *governance framework* sur le triangle de confiance.

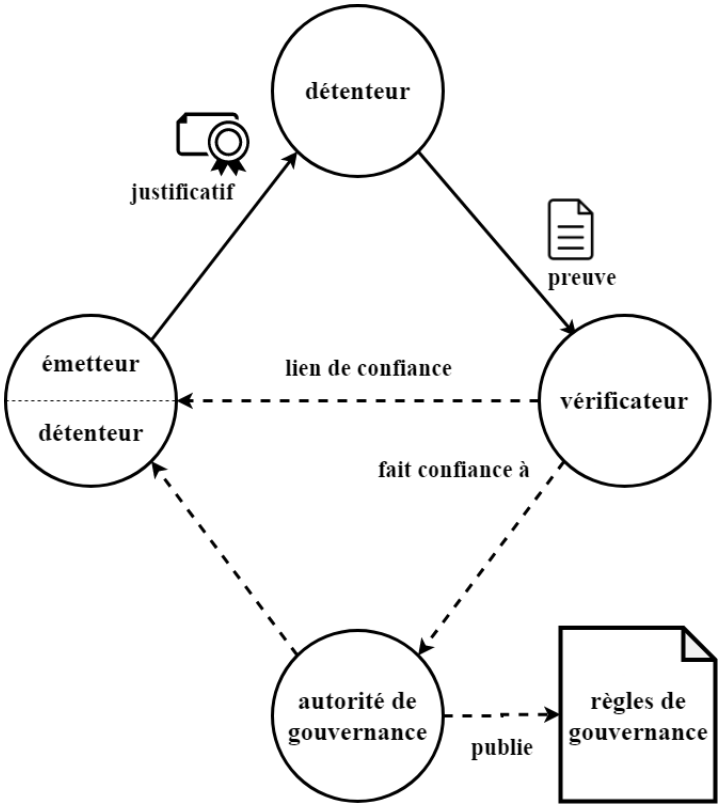


FIGURE 2.28 – *Triangle de confiance avec la Governance Framework*
Source: *de l'auteur inspirée par la figure 2.13 de* PREUKSCHAT *et* REED*, 2021*

Cette autorité de gouvernance permet de simplifier le travail d'un vérificateur. Si un lien de confiance n'est pas établi entre un vérificateur et un émetteur, alors le vérificateur peut s'aider des règles de gouvernance pour vérifier la validité d'un émetteur grâce à, par exemple, une liste blanche d'émetteurs valides (PREUKSCHAT et REED, 2021).

2.1.7 Bénéfices

Nous venons de voir la majorité des blocs qui constituent la *SSI*. Nombreux sont les bénéfices découlant d'une utilisation d'une telle technologie, et nous avons déjà pu avoir un avant-goût de certains de ces bénéfices lorsque nous avons discuté des cas d'utilisation de la *SSI*. Dans l'article, GABRIELLA, TAIJA et PEKKA, 2021, deux catégories principales de bénéfice de la *SSI* ressortent.

Les bénéfices pour la **société** présentent des arguments que nous avons nous-mêmes pu constater au travers de ce document. La *SSI* peut conduire à une transformation digitale, à donner aux individus le contrôle de leurs données et bien évidemment à une meilleure expérience utilisateur en simplifiant les différents processus, par exemple l'authentification (GABRIELLA et al., 2021).

Les bénéfices pour les **organisations** sont divers. La *SSI* peut amener une amélioration dans la sécurité et la confidentialité des produits et services, une promotion de l'innovation, une augmentation des performances financières et, l'un des points les plus importants, à réduire grandement le risque de ***cybercrime*** (GABRIELLA et al., 2021).

Ses dix principes, présentés dans la section 2.1.3, permettent également à la *SSI* d'être conforme au *Règlement général sur la protection des données (RGPD)*, qui est un règlement européen sur le traitement des données personnelles, ainsi qu'à la *Loi fédérale sur la protection des données (LPD)*, la loi suisse sur la protection des données. Cette conformité représente un argument de vente majeur pour son adoption par les différentes sociétés traitant des données personnelles. Avec ces informations, nous pouvons déjà imaginer l'impact que la *SSI* peut avoir sur le monde.

2.1.8 Défis

Malgré les bénéfices apportés par la mise en place de la *SSI*, cette dernière implique également de nombreux défis à résoudre. Dans l'article, GABRIELLA et al., 2021, plusieurs catégories de défis ressortent.

Les défis liés à la **gouvernance** mettent en avant le manque d'un leader dans l'écosystème de gouvernance, ce qui conduit à des entités indépendantes les unes des autres et fixant chacune leurs propres règles. Il est donc difficile de trouver des consensus (GABRIELLA et al., 2021).

Les défis liés à la **technolgie** mettent en avant le manque de maturité de l'écosystème technologique de la *SSI*. Cette immaturité crée une complexité autour de la *SSI* (GABRIELLA et al., 2021). Des problématiques, comme celle concernant la création d'un écosystème *SSI* fonctionnant hors ligne, restent encore sans réponse (PREUKSCHAT et REED, 2021).

D'un point de vue **commercial**, nous remarquons un manque de personnel qualifié pour développer ce genre de technologie. L'implémentation d'un tel système peut également engendrer des coûts importants (GABRIELLA et al., 2021).

L'environnement **légal** et le contexte **réglementaire** sont encore très peu définis. Il est donc difficile de se faire des projections sur la *SSI*. Même si le ***World Wide Web Consortium*** a déjà établi certains standards, comme ceux des *Verifiable Credentials*, il y a toujours un manque de standardisation des différentes technologies (GABRIELLA et al., 2021).

Finalement d'un point de vue **social et organisationnel**, il y a un grand manque de connaissances et de compréhension sur la *SSI*, tout comme des différences d'interprétation sur le modèle même de cette technologie (GABRIELLA et al., 2021). Ces défis ne pourront être résolus que par la collaboration des différents acteurs touchés par l'écosystème de la *SSI*.

2.2 Distributed Ledger Technology

Dans cette section, nous découvrons ensemble ce qu'est la *DLT*. Nous étudions d'abord ses concepts fondamentaux et nous nous concentrons sur une sous-catégorie de la *DLT*, appelée *Blockchain*, pour présenter son fonctionnement. Ensuite, nous nous penchons sur son utilité avec la *Self-Sovereign Identity*. Puis, nous détaillons comment choisir la technologie la plus avantageuse en établissant des critères de comparaisons. Finalement, nous mettons en avant ces résultats pour proposer une recommandation de la technologie la plus adaptée à la *SSI*.

2.2.1 Présentation

L'un des blocs qui constituent la *SSI* est la *DLT*. Il est difficile de donner une définition fixe à ce qu'est la *DLT*, car il n'y en a pas d'officielle. Il est donc possible qu'en lisant des travaux d'acteurs du domaine, la définition utilisée pour parler de *DLT* ou de la *Blockchain* varie entre eux. Nous allons voir ensemble les définitions utilisées pour ces technologies dans de document.

Définitions

La *DLT* est une nouvelle approche technologique sur la manière de stocker et partager ses données à travers plusieurs registres de données appelés en anglais *Distributed Ledger (DL)*. Ceux-ci sont contrôlés par un réseau distribué de serveurs appelés des *nodes*[23]. Ce type de technologie est possible grâce à l'échange en ***peer-to-peer***. Il n'y a pas de serveur central, mais les différents noeuds représentent chacun un serveur et ils communiquent entre eux. (WORLD BANK GROUP, 2017). Le schéma 2.29 montre la différence entre un système d'échanges basé sur un serveur central et un échange en ***peer-to-peer***.

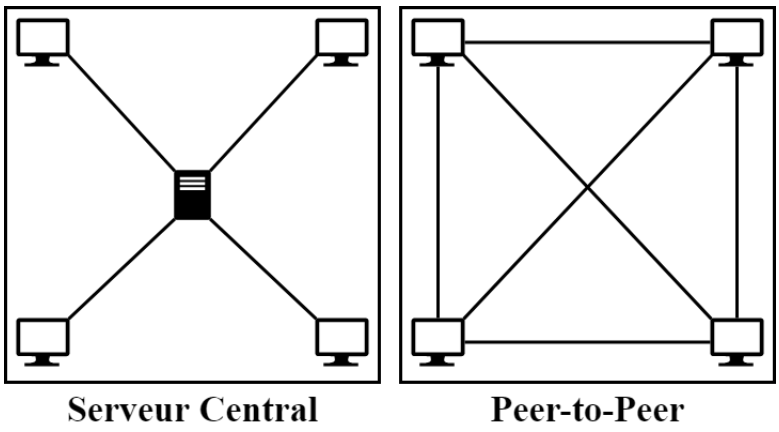


FIGURE 2.29 – *Communication par serveur central et peer-to-peer*
Source: *de l'auteur*

Dans le schéma de droite, nous voyons que les noeuds communiquent directement entre eux.

23. En français, ce sont des *"noeuds"*.

Dans ce travail, nous considérons la *Blockchain* comme un sous-ensemble de la *DLT*. C'est une manière particulière de mettre en pratique la *DLT*. Elle utilise des algorithmes cryptographiques pour la création et la vérification d'une structure de données qui prend la forme d'un bloc, appelé *bloc de transaction*. Chaque bloc possède ce qui s'appelle un ***digest***, qui va être utilisé comme référence par le bloc suivant pour former un lien avec le bloc précédent et ainsi former cette chaîne appelée la *Blockchain*. Le premier bloc se nomme de *Genesis*. Le schéma 2.30 représente la structure d'une *Blockchain* (WORLD BANK GROUP, 2017).

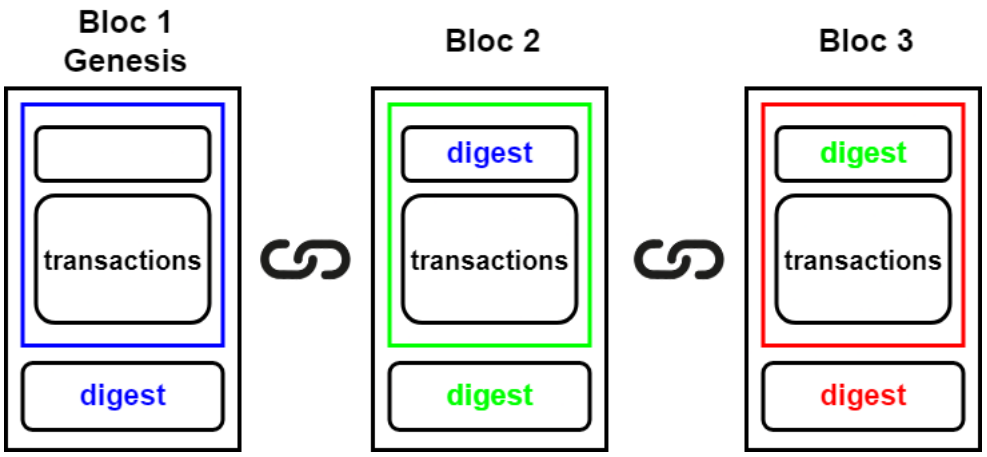


FIGURE 2.30 – *Structure de la Blockchain*
Source: *de l'auteur*

Nous pouvons voir ici comment les différents blocs sont structurés. Tout d'abord, chaque bloc contient une transaction. Cela peut être n'importe quel type de transaction, par exemple un transfert de fonds. Le ***digest*** est généré par une fonction de ***hachage*** prenant en entrée le ***digest*** du bloc précédent ainsi que les transactions du bloc courant. Le bloc *Genesis* ne possède pas le ***digest*** du bloc précédent, car il est le premier bloc de la chaîne.

La *Blockchain*, et plus généralement les *DLs*, permettent au système d'être décentralisé, anonyme, intègre et dont le contenu distribué nécessite un consensus (WORLD BANK GROUP, 2017).

2.2.2 Fonctionnement

Pour comprendre le fonctionnement et l'utilité de ce type de technologies avec la *SSI*, nous allons nous appuyer sur différents concepts de la *Blockchain* Bitcoin. Nous nous sommes basés sur la vidéo 3BLUE1BROWN, 2017 pour expliquer certains des concepts suivants.

Transaction et signature

Imaginons quatre personnes (A, B, C, D) qui effectuent des échanges financiers entre elles. Chaque échange va créer une transaction qui va ensuite être inscrite sur un registre.

Registre des transactions
A verse 10 CHF à B.
B verse 50 CHF à C.
D verse 10 CHF à A.

FIGURE 2.31 – *Exemple d'un registre de transactions*
Source: *de l'auteur*

Ce registre a comme particularité d'autoriser n'importe qui à écrire une transaction. Cependant, pour que ces transactions soient valides, celles-ci doivent être signées par une signature numérique. Comme pour une vraie signature, cette signature numérique évite qu'une action prenne place sans l'accord de la personne concernée. Cela éviterait, par exemple, que B rajoute dix transactions *"A verse 1000 CHF à B"*. Pour que ces transactions soient valides, A devrait les signer.

Avant de continuer, il est important d'expliquer ce qu'est la *cryptographie asymétrique*. La *cryptographie asymétrique* est une méthode qui utilise deux clés liées entre elles, appelées *clé publique* et *clé privée*. La clé publique va permettre de chiffrer un secret, elle peut être communiquée sans problème. La clé privée permet de déchiffrer ce que la clé publique associée a chiffré, elle ne doit donc pas être communiqué, ainsi, si un individu A veut envoyer un message secret à l'individu B, les deux individus vont s'échanger leurs clés publiques. L'individu A va, grâce à la clé publique de B, chiffrer le secret et l'envoyer. À sa réception, l'individu B peut le déchiffrer avec sa clé privée. Le schéma 2.32 illustre ce concept.

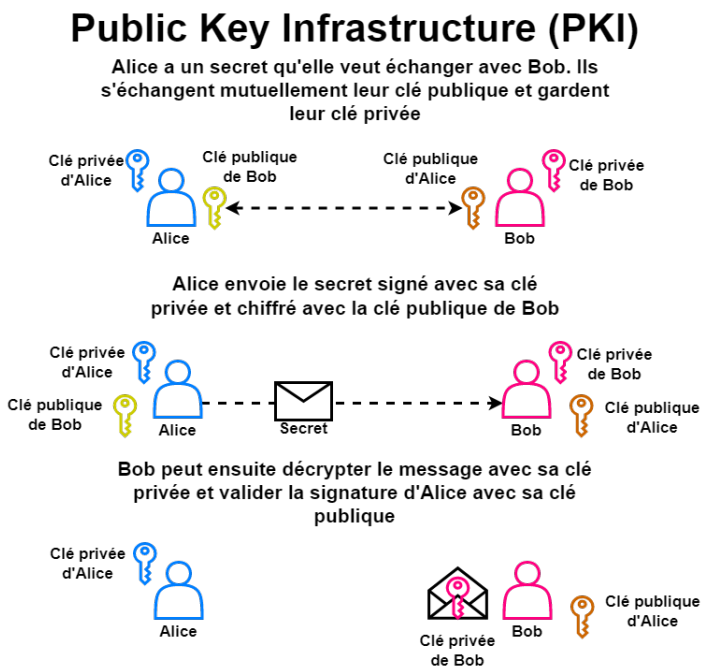


FIGURE 2.32 – *Échange d'un secret avec cryptographie asymétrique*
Source: *traduit et modifié par l'auteur depuis* GISOLFI*, 2018*

Cette signature est construite à partir de l'encryption du message de la transaction par la clé privée.

FIGURE 2.33 – *Génération d'une signature*
Source: *de l'auteur*

Il est ensuite possible pour les autres personnes de vérifier l'authenticité de la transaction grâce à la clé publique.

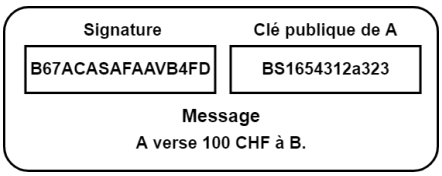


FIGURE 2.34 – *Vérification de la signature d'un message*
Source: *de l'auteur*

Dans ces schémas, les exemples de clés et de signatures sont totalement fictifs. Le but est que, dans un cas réel, la signature générée soit impossible à deviner de par sa complexité de calcul.

Décentralisation, *Proof of Work* et *digest*

Comme nous l'avons vu dans les sections précédentes, la particularité de ce registre est que c'est un *Distributed Ledger*. Pour rappel, cela signifie que celui-ci est distribué. Lorsqu'une nouvelle transaction fait son entrée, celle-ci va être distribuée entre tous les registres. Mais comment savoir quel registre contient les bonnes informations, comme par exemple dans le cas où l'un des registres contient une information différente des autres ?

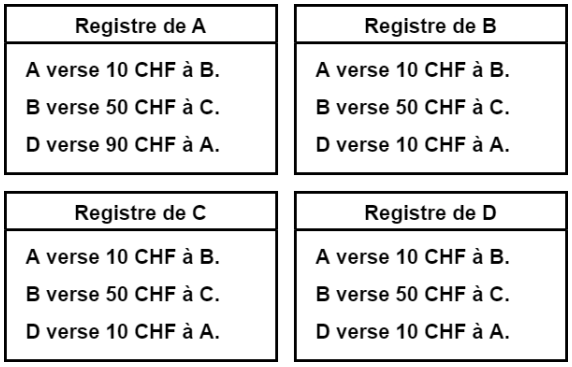


FIGURE 2.35 – *Exemple de registres de transactions avec une différence*
Source: *de l'auteur*

Il faut pour cela ce qui s'appelle une méthode de consensus. Pour le Bitcoin, celle-ci s'appelle la ***Proof of Work***. Par définition, C'est un mécanisme de consensus cryptographique dans lequel un prouveur prouve à des vérificateurs qu'un calcul informatique a été effectué. Nous allons voir en quoi cette preuve consiste exactement mais pour ce faire, nous devons d'abord nous pencher sur ce qu'est une méthode de ***hachage***.

Une méthode de ***hachage*** est une fonction informatique qui va prendre une donnée en entrée et générer, en sortie, une empreinte numérique de taille fixe appelée ***digest***. Pour que celle-ci soit sécurisée, il faut que pour chaque entrée l'empreinte soit différente. De plus, il ne faut pas qu'à partir de cette empreinte numérique il soit possible de retrouver la donnée de base. Une des méthodes de ***hachage*** utilisées est la fonction *SHA-256*. Voilà un exemple d'utilisation de celle-ci.

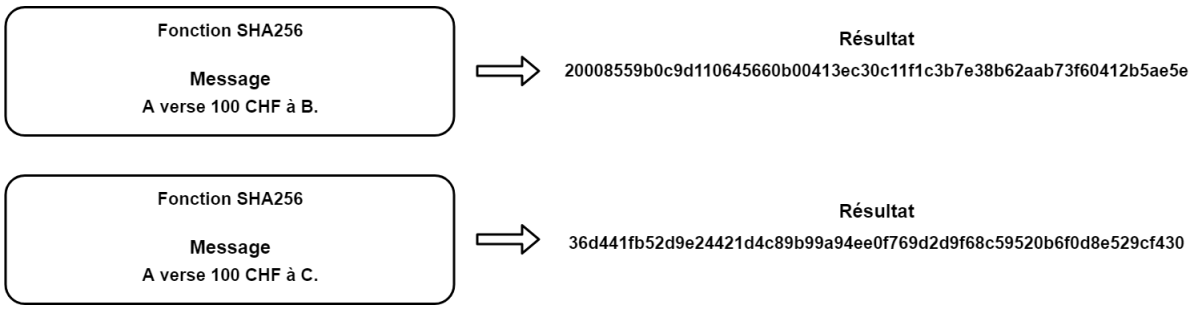


FIGURE 2.36 – *Résultat de la fonction SHA-256 en hexadécimal*
Source: *de l'auteur*

Les résultats présentés sont réels, nous pouvons donc voir qu'un simple changement dans un message, *ici B devient C*, change complètement le résultat. Ici les résultats sont donnés en hexadécimal, il est possible de le convertir en binaire pour obtenir une suite de 1 et 0 de 256 caractères. La fonction *SHA-256* produit ainsi une empreinte numérique de 256 bits.

Ainsi, pour la ***Proof of Work***, l'un des calculs informatiques qu'il est possible de demander est : trouver une valeur nommée ***nonce*** qui, combinée avec la liste des transactions puis passée dans une fonction de ***hachage***, donne une empreinte numérique dont les n premiers bits doivent être constitués de 0. La seule solution pour trouver cette valeur est de tester toutes les possibilités, jusqu'à obtenir la sortie recherchée. Les acteurs qui vont résoudre la ***Proof of Work*** sont appelés des *mineurs* et l'opération de résolution est appelée *minage*. Le schéma suivant illustre ce principe avec 30 premiers bits à 0.

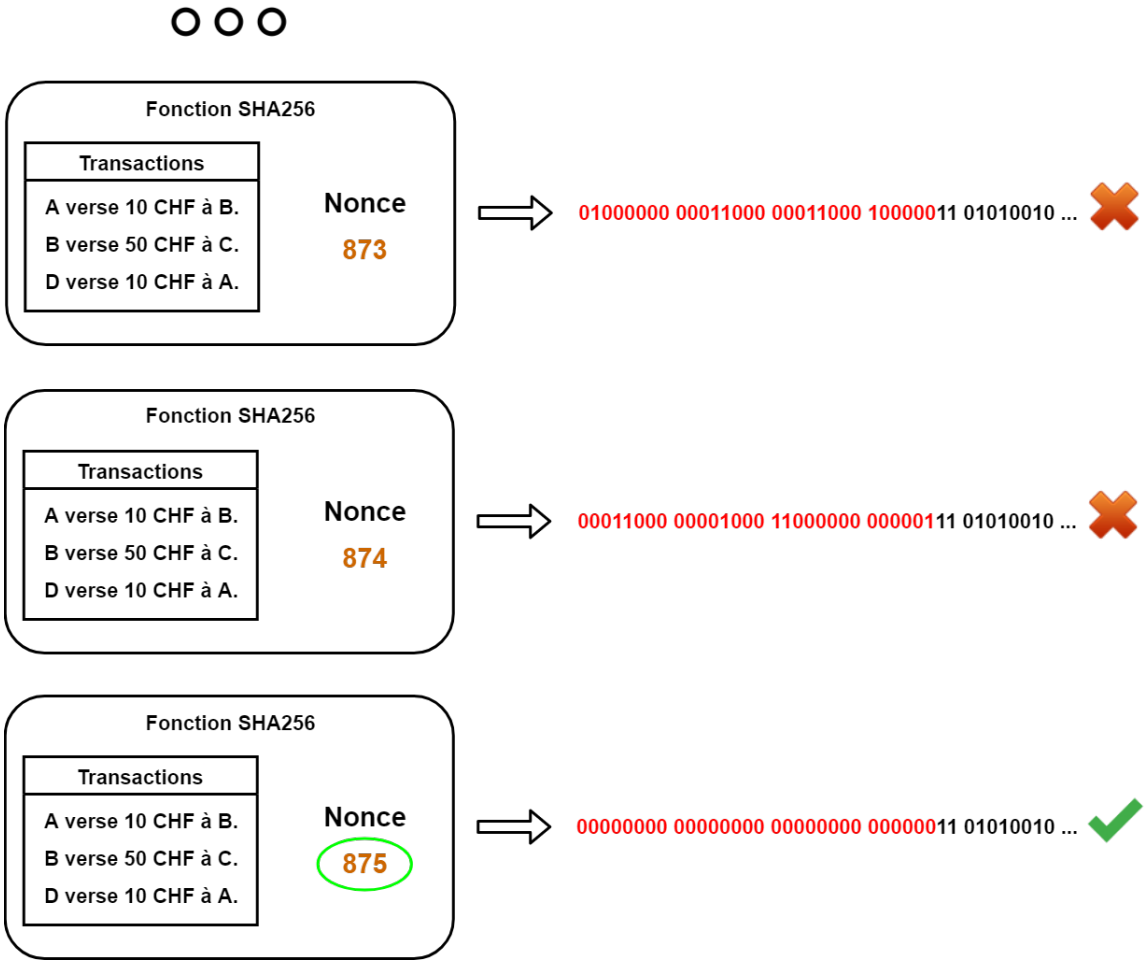


FIGURE 2.37 – *Exemple de calcul de la Proof of Work*
Source: *de l'auteur*

Blockchain

Mais alors pourquoi nous appelons cela la *Blockchain* ? Comme nous l'avons rapidement vu dans la présentation, c'est parce que c'est une chaîne de blocs. Périodiquement, le registre des transactions est inscrit dans un bloc. Pour l'ajouter à la chaîne, il faut d'abord résoudre sa ***Proof of Work***, dont la solution est ensuite ajoutée au contenu du bloc. La totalité du bloc est alors utilisée comme entrée dans une fonction de ***hachage***, et le résultat se trouve dans l'en-tête du bloc suivant pour faire référence au bloc précédent et ainsi former une chaîne.

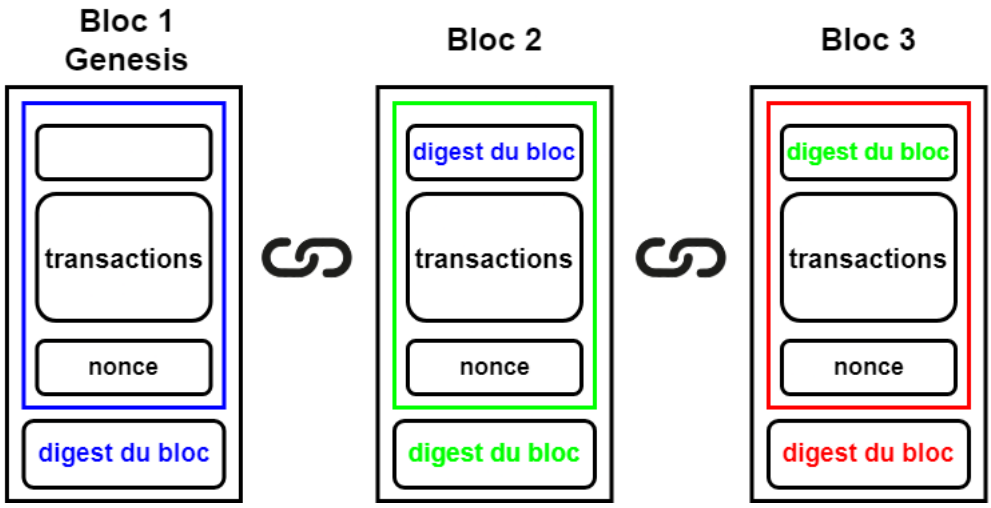


FIGURE 2.38 – *Exemple de Blockchain*
Source: *de l'auteur*

Modification de la Blockchain

Il est possible qu'un acteur malveillant tente de modifier la *Blockchain*. C'est le cas lors de tentatives de ce qui s'appelle le *double-spending*. Comme expliqué dans CHOHAN, 2021, c'est un problème que nous retrouvons dans les systèmes de monnaies numériques et qui consiste à la multiple dépense d'un jeton numérique. Cette opération est possible car un jeton numérique est un fichier qui peut être dupliqué ou falsifié. Par exemple, l'acteur M a 10 CHF sur son compte et verse cette somme à l'acteur A ainsi qu'à l'acteur B. De ce fait, l'acteur M dépense le même jeton, ici les 10 CHF, sur deux transactions différentes.

Cependant, si un acteur tente de modifier la *Blockchain*, une autre version de celle-ci est créée. Comment savoir quelle chaîne est alors la bonne ? Comme nous l'avons dit auparavant, pour qu'un bloc soit ajouté à la chaîne, les mineurs doivent trouver la ***Proof of Work***. Ainsi, si un acteur malveillant est seul, ses ressources sont limitées. Au fur et à mesure que des nouveaux blocs se créent, il devra résoudre pour chacun leur ***Proof of Work*** et les ajouter à sa version de la chaîne. La chaîne principale, disposant de plus d'acteurs, ajoutera alors les blocs beaucoup plus rapidement, créant ainsi une chaîne plus longue. Puisque le protocole *Bitcoin* choisit toujours la chaîne la plus longue, nous pouvons dire qu'il s'agit d'une course de création de blocs entre la fausse chaîne et la chaîne principale (CHOHAN, 2021).

La seule façon pour qu'un acteur malveillant puisse changer la *Blockchain*, et ainsi gagner cette *"course"*, est qu'il contrôle au moins 51% de la puissance de calcul des mineurs sur le réseau. Ce type d'attaque porte le nom de *"51 percent attack"* et est utilisé contre la ***Proof of Work*** (VARSHNEY, 2018). Notons que sur d'autres méthodes de consensus, c'est un pourcentage différent qui doit être contrôlé. Par exemple, la *Delegated Byzantine Fault Tolerance* nécessite le contrôle de plus du tiers des noeuds, par un assaillant, pour une attaque réussie.

Noeuds

Dans le réseau Bitcoin, les participants sont appelés *Noeuds*. Chaque noeud peut avoir différentes fonctions : le ***routage***, la base de données, le minage ou le service de portefeuille. Chaque noeud doit contenir une fonction de ***routage*** pour pouvoir participer au réseau. Notons également que tous les noeuds s'occupent de valider et de propager les transactions et blocs mais également de maintenir la connexion avec les pairs (ANTONOPOULOS, 2014).

Voilà une liste des différents types de noeuds principaux utilisés par Bitcoin et leurs fonctions (ANTONOPOULOS, 2014) :

1. **les noeuds dit *full*** maintiennent une copie à jour de l'entièreté de la *Blockchain*. De ce fait, ils sont capables de vérifier des transactions sans aucune référence externe ;
2. **les noeuds dit *light*** ne contiennent qu'une partie de la *Blockchain*, ils doivent donc utiliser une méthode appelée *simplified payment verification* pour vérifier les transactions ;
3. **les noeuds de minage** sont en compétition pour résoudre la ***Proof of Work*** et ainsi créer un nouveau bloc. Ils peuvent également être *full* et contenir toute une copie de la *Blockchain* ou alors être *Light* et n'avoir qu'une partie de celle-ci.

Il existe également d'autres types de noeuds mais leur explication dépasse la portée de ce travail. Cependant, une présentation détaillée de ceux-ci peut être trouvée dans le chapitre 6 [24] de l'ouvrage ANTONOPOULOS, 2014. La figure 2.39 [25] représente un exemple de l'interaction de ces différents noeuds sur un réseau *Bitcoin*. Chaque carré bleu avec un ou plusieurs ronds à l'intérieur représente un noeud avec une ou plusieurs des quatre fonctions citées auparavant.

Nous venons donc d'avoir un aperçu sur le fonctionnement d'une *Blockchain* et plus particulièrement du Bitcoin. Nous allons maintenant déterminer son utilité avec l'identité autogérée.

24. Ce chapitre est consultable gratuitement à l'adresse suivante.
25. La figure n'est pas traduite car le but n'est pas de comprendre chaque élément mais d'avoir un exemple d'un réseau *Bitcoin*.

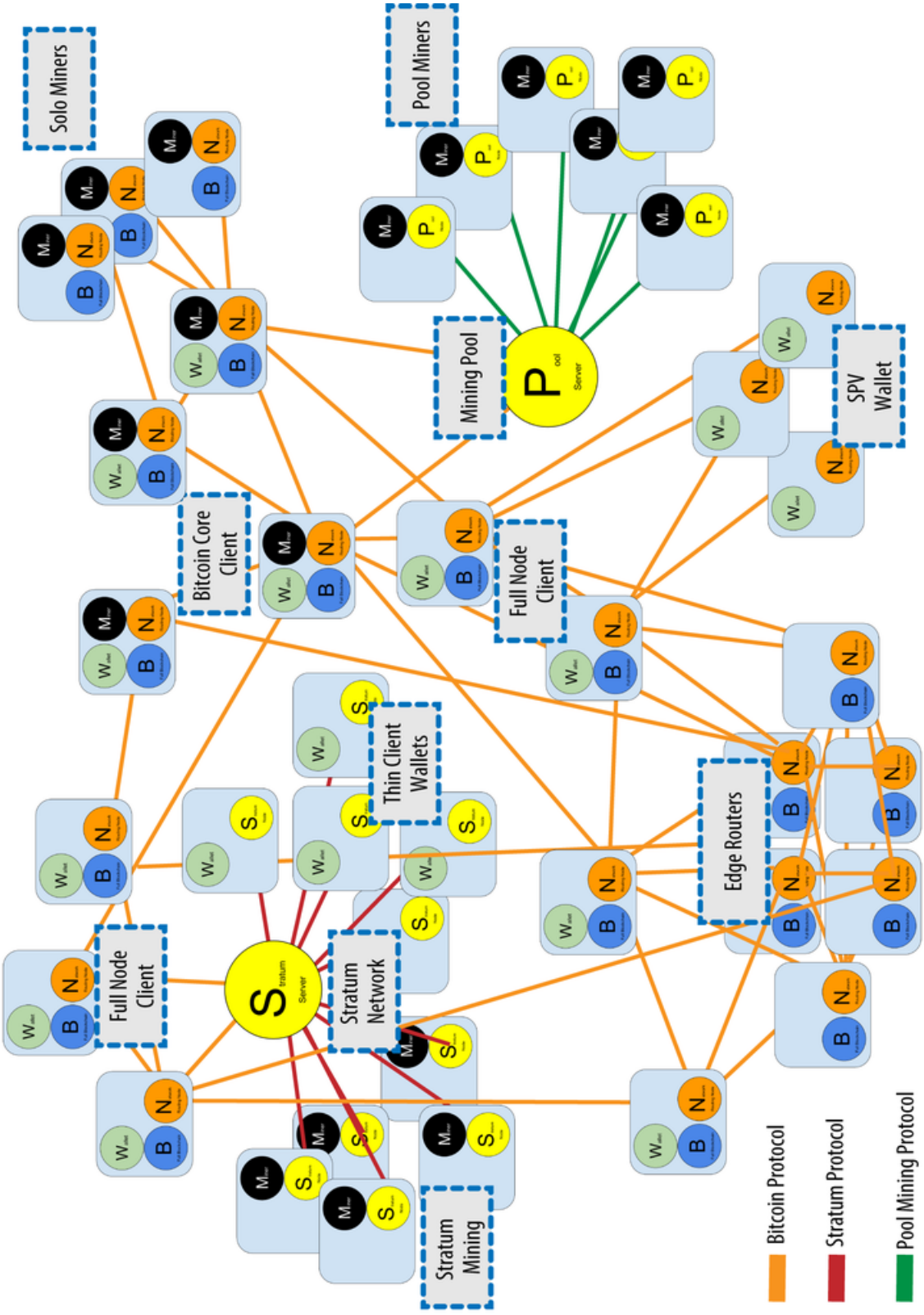


FIGURE 2.39 – *Exemple d'un réseau Bitcoin avec différents noeuds*
Source: ANTONOPOULOS, *2014*

2.2.3 Utilité avec l'identité autogérée

Les *DIDs* ou les *VCs* peuvent être inscrits dans n'importe quel type de registre, même une base de données traditionnelle. Mais alors pourquoi les *DLTs*, et plus particulièrement les *Blockchains* sont-elles intéressantes pour la *SSI* ?

Du fait de son intégrité et de la difficulté à la modifier, la *Blockchain* devient rapidement une source de confiance pour tous les acteurs de la *SSI*. Personne n'a le contrôle sur celle-ci et tous les membres doivent unanimement accepter chaque transaction pour qu'elle soit validée. Aussi, comme leur nom l'indique, les *DLTs* sont distribuées et signées. Cela empêche la corruption des données. La *Blockchain* peut également être programmable grâce à l'utilisation de ***smart contracts***, ce qui permet des traitements personnalisés des transactions (PREUKSCHAT et REED, 2021).

La problématique avec la *cryptographie asymétrique* est qu'il est difficile de vérifier l'authenticité d'une clé publiquement disponible. La *Blockchain* permet cependant la décentralisation de la ***Public Key Infrastructure***, ce qui va répondre à cette problématique (GISOLFI, 2018). Le schéma 2.32 montre un échange sans la décentralisation de l'infrastructure et le schéma 2.40 avec celle-ci. Nous pouvons voir que ce mécanisme décentralisé permet la récupération et la validation de l'authenticité de la clé publique grâce à un registre distribué. Ce sont les règles de gouvernance établies ainsi que la confiance dans les différents noeuds validant le registre distribué qui font que nous pouvons faire confiance au *DID* d'Alice.

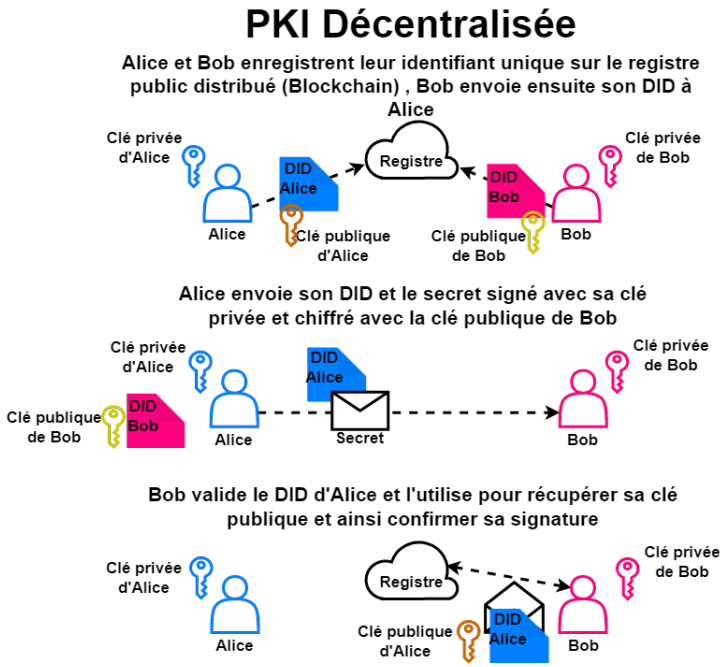


FIGURE 2.40 – *Échange d'un secret avec cryptographie asymétrique décentralisée*
Source: *traduit et modifié par l'auteur depuis* GISOLFI, *2018*

Voilà certaines raisons qui font que la *Blockchain* est un choix intéressant comme registre distribué pour la *SSI*.

2.2.4 Choisir sa technologie

La communauté semble se tourner vers la *DLT*. Plus de 90% des méthodes *DID* définies par le standard du ***World Wide Web Consortium*** sont basés sur des technologies distribuées (PREUKSCHAT et REED, 2021). Nous présentons, tout d'abord, ces technologies et leurs caractéristiques clés. Ensuite, nous les regroupons ensemble dans un tableau comparatif, pour finalement pouvoir proposer la technologie la plus adaptée à la *SSI*.

Ethereum

Ethereum apparaît pour la première fois dans un article [26] écrit en 2013 par Vitalik Buterin, considéré aujourd'hui comme le fondateur d'Ethereum. En 2014, la technologie a le droit à une définition technique [27] définie par Gavin Wood. C'est en 2015 que le projet est officiellement lancé (ETHEREUM, s. d.-a). Ethereum est une *Blockchain* publiquement accessible, décentralisée, ***open source*** et qui permet l'utilisation des ***smart contracts***. Elle est utilisée dans de nombreux domaines comme la finance ou le commerce. Le 11 juin 2022, le prix pour un Ethereum est de 1630 dollars, est constitué de 2047 noeuds et il y a plus d'un million de transactions par jour (ETHEREUM, s. d.-a).

Ethereum se présente comme décentralisé sans l'être complètement. Son développement est supervisé et modelé par Vitalik Buterin. Le projet est également supporté par la Fondation Ethereum qui siège en Suisse. Il n'y a pas non plus de mécanisme de consensus pour accepter de nouveaux changements au protocole. Pour exécuter sa *Blockchain*, Ethereum utilise un regroupement de code appelé *geth* ce qui, si celui-ci est compromis ou vulnérable, met toute la *Blockchain* en danger. Nous pouvons également ajouter que sa distribution géographique est également centralisée avec un pourcentage de 41% des noeuds situés aux États-Unis (AGARWAL, 2022).

La confidentialité sur Ethereum est supportée grâce à l'utilisation de la *ZKP*, notons cependant que celle-ci n'est pas supportée en production. Les deux types de technologies de *ZKPs* utilisées sont les *zk-SNARKs* et les *zk-STARKs* (ETHUB, s. d.). La version actuelle (1.0) d'Ethereum utilise un mécanisme de consensus appelé ***Proof of Work*** mais cette méthode est très largement critiquée à cause de son impact environnemental. Par année, Ethereum consomme l'équivalent de la consommation électrique du Chili et émet une empreinte carbone comparable à celle de Hong Kong (DIGICONOMIST, s. d.-b). L'Ethereum veut cependant passer à une version 2.0 et remplacer la ***Proof of Work*** par la ***Proof of Stake***. Ce changement a pour but de réduire fortement la consommation énergétique de la technologie mais également de la rendre plus décentralisée (AGARWAL, 2022).

Finalement, le prix d'une transaction au mois de juin 2022 se situe autour de 2 dollars mais a dans le passé atteint plus de 60 dollars (YCHARTS, s. d.-b). Le temps de création d'un bloc peut lui monter jusqu'à 14 secondes (ETHEREUM, s. d.-a).

26. Consultable à l'adresse *https://ethereum.org/en/whitepaper/*.
27. Consultable à l'adresse *https://github.com/ethereum/yellowpaper*.

ION

ION est un réseau décentralisé qui s'exécute sur la *Blockchain* Bitcoin. Son nom complet est *Identity Overlay Network* qui signifie *réseau de superposition d'identités*. Il a été conçu spécifiquement pour l'élaboration d'identités décentralisées (IDENTITY FOUNDATION, s. d.-a).

C'est la société *Microsoft*, en partenariat avec le ***World Wide Web Consortium***, qui a majoritairement financé et conduit le projet de création d'ION (SIMONS, 2019). Cependant, ce projet "appartient" à la *Decentralized Identity Foundation (DIF)*. Cette fondation a pour but de regrouper des membres internationaux pour collaborer à la création d'un écosystème décentralisé (IDENTITY FOUNDATION, s. d.-a).

Il est difficile d'estimer l'impact de *Microsoft* sur ION, cependant la fondation nous assure que même si celui-ci est un investisseur important, son rôle ne s'étend pas au-delà de celui de sponsor (IDENTITY FOUNDATION, s. d.-b). La *Blockchain* Bitcoin, sur laquelle ION est construit, est considérée comme la plus décentralisée de toutes (AGARWAL, 2022).

ION est publiquement accessible et son code source est consultable gratuitement et librement [28]. Cette technologie se base également sur le protocole *Sidetree* qui est également ***open source*** [29]. Le protocole *Sidetree* permet la création d'un réseau de *DIDs* qui peuvent s'exécuter sur n'importe quel système décentralisé comme Ethereum ou Bitcoin. Cependant, il est important de noter qu'une technologie utilisant ce protocole dépend du système sur lequel *Sidetree* s'exécute, pour ION il s'agit du Bitcoin (IDENTITY FOUNDATION, s. d.-a).

Il n'y a pas de mécanisme de consensus pour ION car, selon la documentation officielle, les noeuds traitent les opérations selon un ensemble de règles déterministes, sans nécessiter un mécanisme de consensus séparé. La technologie supporte l'utilisation de *ZKPs* ainsi que de ***smart contracts*** (IDENTITY FOUNDATION, s. d.-b).

Bitcoin a un lourd impact écologique. Il consomme deux fois plus d'électricité qu'Ethereum et a une empreinte carbone trois fois plus importante (DIGICONOMIST, s. d.-a). Cela est dû au mécanisme de ***Proof of Work***. Toutefois, même si ION n'utilise pas de mécanismes de consensus, il est construit sur Bitcoin et partage alors sa consommation énergétique. La documentation ajoute également qu'un noeud ION, indépendamment du Bitcoin, ne consomme rien de plus que du stockage.

Finalement, le prix d'une transaction au mois de juin 2022 se situe autour de 1.10 dollars, mais a atteint au mois de mai 2022 plus de 20 dollars (YCHARTS, s. d.-a). La documentation de ION nous dit cependant que plus de 10'000 opérations peuvent être contenues dans une seule transaction (IDENTITY FOUNDATION, s. d.-b). Pour ce qui est du temps de création d'un nouveau bloc, celui-ci peut prendre de 20 minutes à 2 heures (MATTR LEARN, s. d.-a).

28. Consultable à l'adresse *https://github.com/decentralized-identity/ion*.
29. Consultable à l'adresse *https://github.com/decentralized-identity/sidetree*.

IOTA

IOTA est à la fois une *DLT* et un protocole de transfert. Il a été conçu pour permettre l'échange sécurisé de données. La technologie permet également une intégrité des données ainsi qu'une conformité au *RGPD* (IOTA SERVICES, s. d.).

C'est la fondation IOTA qui supervise le projet. Elle siège à Berlin et a un but non lucratif. La fondation est également présente dans plus de 25 pays. Le financement provient majoritairement des dons de la communauté, d'organismes publics ou de contributions privées. Le conseil d'administration de la fondation s'occupe de donner des conseils et s'assure que la vision soit bien atteinte (IOTA, s. d.-b).

IOTA est accessible publiquement. Son code source est également consultable gratuitement et librement [30] sur la plateforme *GitHub* (IOTA, s. d.-b). Du fait de son architecture et de son fonctionnement, IOTA n'est pas décentralisé. Cependant, une version 2.0 est en développement promettant un *Distributed Ledger* décentralisé, sécurisé et entièrement gratuit (IOTA, s. d.-a).

IOTA utilise un mécanisme de consensus spécifique à son écosystème. Ce consensus est possible car IOTA ne se base pas sur le modèle de *Blockchain* traditionnel mais sur celui d'un concept appelé *Tangle*. Nous n'entrerons pas dans les détails de son fonctionnement car cela dépasse la portée de ce document. Toutes les explications techniques sont disponibles sur le Wikipedia officiel de la fondation. IOTA supporte également les ***smart contracts*** et les ***Zero-Knowledge Proofs (ZKP)*** (IOTA WIKI, s. d.).

IOTA est considéré comme une *DLT* verte. La technologie ne consomme pas beaucoup d'énergie. Lors d'une étude réalisée, pour 100'000 transactions, le système de paiement *MasterCard* consomme environ 70 kilowatt-hour contre 11 pour IOTA (SORI ABBASZADEH, 2019). De plus, la fondation est impliquée dans plusieurs projets de préservation de l'environnement (IOTA, s. d.-b). Finalement, les transactions avec IOTA sont gratuites. Le temps de transaction est lui d'environ 10 secondes (IOTA SERVICES, s. d.).

30. Consultable à l'adresse https://github.com/iotaledger.

Partisia

Partisia est une *Blockchain* dont le développement est axé sur la confiance, la transparence ainsi que le respect de la vie privée. C'est une *Blockchain* de la couche 1 utilisée pour le traitement public et privé (PARTISIA BLOCKCHAIN, s. d.-b).

Le projet est supervisé par la fondation Partisia qui a comme objectif de supporter la *Blockchain* publique. La fondation est située à Zoug, en Suisse, et est financée par des privés. En juin 2022, 28 développeurs travaillent sur le projet et la *Blockchain* a plus de 50 applications commerciales (PARTISIA BLOCKCHAIN, s. d.-b).

Le code source de la *Blockchain* n'est, pour l'instant, pas disponible publiquement mais il est possible de l'obtenir en contactant l'équipe de développement par courriel (PARTISIA BLOCKCHAIN, s. d.-a). Partisia supporte différents actifs pour son écosystème, par exemple l'Ethereum ou le Bitcoin. Au 14 juin 2022, il y a plus de 180'000 blocs et 80 validateurs à travers le monde (PARTISIA BLOCKCHAIN, s. d.-b).

Partisia propose un mécanisme de consensus modulaire, ce qui signifie qu'il est possible de choisir le protocole utilisé selon les besoins (PARTISIA BLOCKCHAIN, 2021). La *Blockchain* supporte également les ***smart contracts*** ainsi que l'utilisation de la ***ZKP*** (PARTISIA BLOCKCHAIN, s. d.-a).

Du fait de pouvoir choisir son mécanisme de consensus, l'impact écologique de cette *Blockchain* peut varier. Il n'y a pas de chiffres sur sa consommation électrique ou son émission de gaz carbonique. Cependant, la technologie se présente comme verte et supporte plusieurs projets de préservation de l'environnement (PARTISIA BLOCKCHAIN, s. d.-b).

Finalement, Partisia a un temps moyen de création de blocs de 3 secondes et le coût d'une transaction se monte à 0.01 dollar (PARTISIA BLOCKCHAIN, s. d.-b).

Sovrin

Sovrin est une fondation américaine à but non lucratif. Elle utilise, pour construire son réseau, une base construite sur *HyperLedger Indy* (SOVRIN, s. d.). C'est un *Distributed Ledger* qui a été créé dans le but de construire des identités décentralisées (HYPERLEDGER INDY, s. d.). Il faut donc, pour analyser cette proposition, nous intéresser à Sovrin et Indy.

L'organisme qui supervise le projet est la fondation Sovrin. Les décisions concernant la fondation sont sont prises par un comité de huit personnes et les décisions sur les aspects techniques de Sovrin sont prises par un comité de 11 personnes. Sovrin propose trois types de réseaux différents mais tous basés sur *Hyperledger Indy*, contenant chacun entre 4 et 25 noeuds opérés par ce que la fondation appelle des *Stewards*. Ces *Stewards* sont des organisations privées [31]. La fondation sert alors uniquement d'autorité de gouvernance dont le but est de permettre au mieux l'implémentation d'une identité décentralisée (SOVRIN, s. d.).

Notons que dans le comité prenant les décisions sur la fondation, la plupart d'entre eux sont résidents aux États-Unis. La fondation étant également inscrite aux États-Unis, il est possible que celle-ci doive dans certains cas collaborer avec les autorités américaines. Sovrin ne répond pas à la conformité du *Règlement général sur la protection des données* (DÉLÈZE IVAN, 2020).

Le code source d'Indy est disponible librement [32]. L'accès à Sovrin est public mais l'écriture sur ses réseaux est limitée selon le type de réseau choisi. Sovrin propose trois réseaux différents (SOVRIN, s. d.).

1. ***BuilderNet*** est un réseau pour le test et le développement. Il n'y a pas de coûts de transactions mais le réseau n'est valide que 6 mois sans pouvoir être renouvelable.
2. ***StagingNet*** est un réseau pour la préproduction. Sa mise en place coûte 500 dollars pour 6 mois et est non renouvelable. En plus, des coûts de transactions s'appliquent, allant de 0.01 dollar pour les mises à jour à 5 dollars pour l'écriture de schémas.
3. ***MainNet*** est un réseau pour la production. Sa mise en place coûte 5000 dollars et est valable une année, avec fonction de renouvellement. Il y a également des coûts de transactions allant de 0.10 dollar pour les mises à jour jusqu'à 50 dollars l'écriture de schémas.

Tous ces réseaux permettent également l'utilisation de ***smart contracts*** ainsi que de ***ZKPs*** (HYPERLEDGER INDY, s. d.). Le mécanisme de consensus utilisé pour *Hyperledger Indy* est appelé *RBFT* (HYPERLEDGER, s. d.). Nous n'avons pas trouvé de chiffres sur l'impact écologique de Sovrin et d'*Indy*, mais nous savons que la fondation Hyperledger est engagée pour la défense du climat. Il n'y a pas non plus d'informations sur le temps de création d'un bloc.

31. *Swisscom*, le géant de la télécommunication suisse, est l'un de ces *Stewards*.
32. Consultable à l'adresse *https://github.com/hyperledger/indy-node*.

Comparaison

Nous venons de faire la présentation de diverses technologies. Avant de faire une recommandation sur la *DLT* la plus adaptée, nous allons faire une comparaison de ces technologies. Pour ce faire, les tableaux 2.2 récapitulent les différents critères des technologies précédemment présentées.

Critère / Technologie	**Ethereum**	**ION**	**IOTA**
Prix (transaction)	~2 dollars	~1.10 dollars	Gratuit
Temps de création (bloc)	~13 secondes	~20 minutes à 2 heures	~10 secondes
Support *smart contracts*	Oui	Oui	Oui
Support *ZKP*	Oui	Oui	Oui
Organisation	Ethereum Foundation (CH)	*DIF* (USA)	IOTA Foundation (DE)
Degré décentralisation	Moyen	Haut	Centralisée
Accès public	Oui	Oui	Oui
Code *open source*	Oui	Oui	Oui
Méthode de consensus	***Proof of Work***	Aucune	Spécifique
Impact écologique	Haut	Haut	Bas

Critère / Technologie	**Partisia**	**Sovrin**
Prix (transaction)	0.01 dollar	0 à 50 dollars
Temps de création (bloc)	~3 secondes	Inconnu
Support *smart contracts*	Oui	Oui
Support *ZKP*	Oui	Oui
Organisation	Partisia Foundation (CH)	Sovrin Foundation (USA)
Degré décentralisation	Haut	Bas
Accès public	Oui	Limité
Code *open source*	Prévu	Non
Méthode de consensus	Modulable	RBFT
Impact écologique	Bas	Inconnu

TABLE 2.2 – *Tableaux de comparaison de différentes DLTs*
Source: *de l'auteur*

2.2.5 Recommandation

Pour pouvoir faire une recommandation adéquate de la *DLT* à utiliser, il est important de se remémorer la problématique. Le Département fédéral de justice et police doit trouver une solution pour pouvoir fournir une ***e-ID*** à la population suisse.

Cette solution doit donc répondre à plusieurs critères que nous avons établis précédemment. Il est par exemple nécessaire que la technologie utilisée soit décentralisée, car il n'est pas acceptable que la population perde le contrôle de ses données. Ainsi, Ethereum, Sovrin et IOTA sont des technologies qui, dans leur version actuelle, ne répondent pas à ce critère fondamental, elles ne sont donc pas envisageables pour la mise en place d'une ***e-ID***.

Un autre critère important est l'impact environnemental de la *DLT*. ION étant construit sur la *Blockchain* Bitcoin, celle-ci a un impact massif sur l'environnement. De plus, la fondation *DIF*, qui gère le projet, siège aux États-Unis, ce qui peut être un frein pour la mise en place d'un ***e-ID*** en Suisse, à cause de l'***USA PATRIOT Act*** (IDENTITY FOUNDATION, s. d.-a). Notons que c'est également le cas pour Sovrin, dont la fondation siège aux États-Unis (SOVRIN, s. d.).

Le dernier choix se porte alors vers Partisia. Celle-ci répond aux critères demandés. Le coût de transaction de Partisia est de 0.01 dollar par transaction. Par rapport à une carte bancaire, où il est parfois question de 2.5% de la valeur de la transaction, c'est négligeable.

Elle est également verte et investie dans la protection de l'environnement ainsi que sur divers aspects sociaux, comme la diversité et l'inclusivité. Son degré de décentralisation est haut et elle est disponible publiquement.

Finalement, nous recommanderions de choisir Partisia car elle présente toutes les caractéristiques recherchées et la fondation qui l'entoure, la fondation Partisia, se trouve en Suisse. Ce qui, pour une proposition générale à la population, peut servir à renforcer la confiance envers cette technologie.

2.3 Quelques solutions existantes

Dans cette section, nous présentons différentes solutions pour la *Self-Sovereign Identity* actuellement disponibles sur le marché. Certaines spécificités techniques et certains concepts, relatifs à ces solutions, sont hors du cadre de ce travail et leur explication ne ferait que complexifier la compréhension générale du sujet. C'est pourquoi nous nous concentrons sur leurs caractéristiques principales, leurs avantages et leurs inconvénients.

2.3.1 Hyperledger Aries

Présentation

Aries est une bibliothèque maintenue par la fondation Hyperledger qui est soutenue par la fondation Linux. Elle se présente comme un kit d'outils pour la création, la transmission et le stockage de *Verifiable Credentials*. Elle est construite pour permettre les interactions basées sur la *Blockchain* en ***peer-to-peer*** (HYPERLEDGER FOUNDATION, s. d.-d).

La documentation principale du projet se trouve sur la plateforme GitHub. Mais la fondation met également à disposition un Wiki, ainsi que des cours de formation sur *Aries* : "Becoming a Hyperledger Aries Developer", "Introduction to Hyperledger Sovereign Identity Blockchain Solutions: Indy, Aries & Ursa".

Aries est un projet ***open source*** qui est mis gratuitement et librement à disposition. La communauté est active sur la plateforme de communication *Discord*, ce qui permet de poser des questions en temps réel aux acteurs impliqués dans le projet. Notons que le projet est encore en développement et que la documentation peut parfois être incomplète ou dépassée. De ce fait, cela rend l'entrée d'un nouveau venu, dans l'univers *Aries*, compliquée.

Caractéristiques

Il existe plusieurs versions de la bibliothèque et toutes ont leurs propres fonctions et spécificités (HYPERLEDGER FOUNDATION, s. d.-a). Voyons d'abord une liste de caractéristiques communes entre les différentes versions de la bibliothèque (HYPERLEDGER FOUNDATION, s. d.-e).

1. Une couche d'interface dédiée, appelée *Resolver*, permet de créer, signer et lire des transactions avec la *Blockchain*.
2. Un stockage cryptographique est à disposition pour le stockage de secrets, de ***Verifiable Credentials (VC)*** et d'autres informations utiles pour le développement de clients dédiés à l'échange de ***VC***.
3. Un système ***peer-to-peer*** chiffré, appelé *DIDComm*, permet les interactions entre les clients en se basant uniquement sur les *DIDs*.
4. Un support pour l'échange de différents formats de ***VC***, par exemple avec l'utilisation de *ZKPs*, est à disposition des développeurs.

5. Une série de protocoles permet l'implémentation et le déploiement des agents [33].
6. Des exemples d'implémentations, selon des cas d'utilisation spécifiques, prêts à la production sont disponibles.
7. Un module dédié permet de tester en continu les agents développés avec Aries.

Avant de passer à une présentation de certaines versions de la bibliothèque, il est important de mentionner que leur architecture est composée d'un ***framework*** et d'un contrôleur. Le ***framework*** va permettre à l'agent numérique d'interagir avec les *Distributed Ledgers*, le stockage et les autres agents. Elle permet également d'initialiser les connexions, répondre aux requêtes ou encore d'envoyer des messages. Cependant, le ***framework*** ne sait pas, par exemple, quand et quoi répondre à la requête, c'est pourquoi il y a un contrôleur. Le contrôleur a pour but de contrôler le comportement du ***framework***, c'est-à-dire définir, selon notre cas d'utilisation, les règles métiers à appliquer lors de différents événements (LINUX FOUNDATION, s. d.). L'architecture générale, de la plupart de ces version, peut alors être représentée selon le schéma suivant.

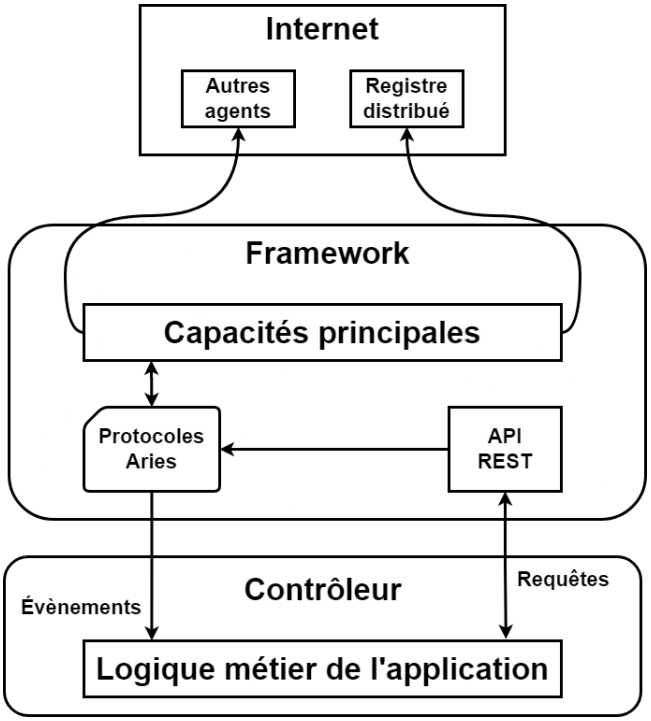


FIGURE 2.41 – *Architecture de la bibliothèque Aries*
Source: *traduit et modifié par l'auteur depuis* HYPERLEDGER FOUNDATION*, s. d.-f*

Nous pouvons donc voir que le contrôler va permettre, grâce à la logique métier, de répondre aux évènements reçus mais également d'envoyer des requêtes en passant par une interface de programmation ***REST***. Le ***framework*** va donc traiter ses requêtes avec les capacités principales implémentées dans celui-ci. C'est également lui qui va communiquer avec les autres agents et avec le *Distributed Ledger*.

33. Un agent, dans l'écosystème Hyperledger, désigne un logiciel qui va interagir avec les autres entités en utilisant des *DIDs*, par exemple.

Aries Cloud Agent Python

Cette version du ***framework*** est utilisée pour le développement d'agents non mobiles. Elle et son contrôleur s'exécutent en parallèle et communiquent grâce à une ***Application Programming Interface (API) REST*** comme dans le schéma 2.41. Elle va s'exécuter sur un serveur et non sur un périphérique mobile (HYPERLEDGER FOUNDATION, s. d.-f).

Le ***framework*** est développé avec le langage Python, mais le contrôleur peut être programmé avec le langage désiré, tant que ce langage permet de communiquer avec une ***API REST***. Notons également que le ***framework*** permet d'utiliser les protocoles *AnonCred* ainsi que *JSON-LD avec BBS+*[34] pour délivrer, vérifier et stocker les *Verifiable Credentials* (HYPERLEDGER FOUNDATION, s. d.-f).

Un point important à mentionner avec cette version est que celle-ci ne permet pas d'utiliser n'importe quel *Distributed Ledger* : elle est liée à *Hyperledger Indy*[35]. Le but de la fondation Hyperledger est de rendre ce ***framework*** *Blockchain agnostic*, c'est-à-dire que n'importe quel *DL* puisse être utilisé (HYPERLEDGER FOUNDATION, s. d.-f).

Le ***framework*** est mis à jour régulièrement mais celui-ci permet déjà le développement et la production de solutions complètes pour la *SSI*. C'est la bibliothèque *Aries* avec la communauté la plus active (HYPERLEDGER FOUNDATION, s. d.-f).

Aries Framework Dotnet

Cette version a pour but de fournir une bibliothèque pour le développement de solutions pour la *SSI* pour le ***Cloud***, les périphériques mobiles ainsi que pour l'*Internet of Things*.

Au même titre que l'*Aries Cloud Agent Python*, ce ***framework*** permet de développer le contrôleur dans le langage de notre choix mais est également limité sur le choix du *DL* à utiliser (HYPERLEDGER FOUNDATION, s. d.-b).

Aries Framework Go

La dernière version que nous présentons est celle implémentée avec le langage Go. Elle est encore en développement mais suscite déjà l'intérêt dans la communauté Hyperledger. Elle se présente comme une version totalement indépendante d'*Aries*, c'est-à-dire que celle-ci ne dépend d'aucune autre bibliothèque externe. De plus, elle est entièrement codée avec le langage Go (HYPERLEDGER FOUNDATION, s. d.-c).

Même si celle-ci est encore en développement, l'objectif de cette bibliothèque est d'offrir un kit d'outils pour l'échange de *DIDs* et de *VC*, une solution de communication via le protocole *DIDComm* et surtout une implémentation *blockchain agnostic* d'Aries. Pour ce faire, le ***framework*** a

34. L'explication de ces deux protocoles dépasse la portée de travail.
35. C'est un *Distributed Ledger* développée par Hyperledger.

un modèle dit de *batteries incluses*, c'est-à-dire que des composants par défaut sont inclus pour réaliser toutes ses actions, mais qu'il est possible de les modifier avec d'autres composants adaptés à chaque situation (HYPERLEDGER FOUNDATION, s. d.-c).

Notons également qu'il y a deux façons de travailler avec ce ***framework***. La première consiste à l'exécuter indépendamment et de développer les contrôleurs dans le langage de notre choix, comme la version *Aries Cloud Agent Python*. La deuxième façon de l'utiliser est de développer l'entièreté des composants en Go pour avoir une solution adaptée au besoin métier (HYPERLEDGER FOUNDATION, s. d.-c).

Cette version dispose d'une communauté active et de plusieurs exemples d'implémentation selon des cas d'utilisation différents. Cependant, du fait de son développement actif, certains exemples et guides techniques sont dépassés (HYPERLEDGER FOUNDATION, s. d.-c).

Avantanges

Les avantages de cette solution sont multiples. Tout d'abord, celle-ci est entièrement gratuite et ***open source***. De plus, la communauté autour du projet est active, ce qui permet d'avoir rapidement une réponse à ses questions. La version du ***framework*** en Python est bien maintenue et testée. De nombreux exemples d'implémentations sont également trouvables sur Internet. La version en Go actuelle offre le fait d'être *blockchain agnostic*, ce qui est une caractéristique recherchée pour le développement de solutions pour la *SSI*.

La liberté de choisir le langage de programmation pour le développement du contrôleur, ainsi que l'utilisation d'une ***API REST***, permettent aux développeurs de mettre en place rapidement cette bibliothèque sans avoir besoin d'apprendre de nouveaux langages, ce qui pourrait représenter un investissement trop important de temps.

Inconvénients

Le problème le plus important de cette solution est que si nous souhaitons utiliser une version stable, comme la version en Python, nous sommes obligés d'utiliser les composants implémentés par le ***framework***, comme le *Distributed Ledger*. Notons que la fondation Hyperledger cherche à corriger ce problème pour offrir des implémentations flexibles de leurs bibliothèques. Il ne faut pas non plus oublier que la plupart des versions d'*Aries* sont encore en développement, des fonctionnalités peuvent alors manquer ou changer complètement d'une version à une autre.

Un autre problème que nous avons rencontré durant l'écriture de ce travail est que même si la communauté est active, il est parfois compliqué d'obtenir des réponses à nos questions si nous sommes un novice. De plus, comme mentionné auparavant, certaines démonstrations, ainsi que la documentation, sont parfois dépassées. Cela peut être un frein pour l'implémentation de cette solution, car le temps investi dans la compréhension du fonctionnement devient alors trop important.

2.3.2 MATTR

Présentation

MATTR est une société néo-zélandaise qui a pur but d'offrir à ses clients des outils pour construire une confiance numérique. Pour ce faire, elle propose une solution appelée *MATTR VII* (*MATTR Seven*). C'est un ensemble d'interfaces qui vont permettre la gestion de *VCs* et de *DIDs* (MATTR, s. d.).

Notons qu'elle propose également un kit de développement logiciel, basé sur sa solution *MATTR VII*, pour permettre aux développeurs d'implémenter une solution personnalisée pour la *SSI*. Nous allons uniquement nous concentrer sur les caractéristiques de *MATTR VII*, puisque cette solution offre un plus grand panel de fonctionnalités que le kit de développement (MATTR, s. d.).

La solution *MATTR VII* n'est ni ***open source***, ni gratuite. Il est cependant possible de la tester, sans aucun frais mais avec des limitations. La société dispose d'un dépôt sur GitHub et est très présente sur les réseaux sociaux comme Youtube ou encore Twitter (MATTR, s. d.).

Une documentation complète et détaillée peut être trouvée sur la chaine Youtube de la société ainsi que sur leur plateforme d'apprentisage. Ces ressources représentent une aide importante et utile pour tous les nouveaux développeurs souhaitant utiliser les services de *MATTR* (MATTR, s. d.).

Caractéristiques

Nous allons nous concentrer sur l'architecture de la solution *MATTR VII*. Les composants principaux proposés par la solution sont au nombre de quatre (MATTR LEARN, s. d.-b).

1. **VII Core** est le composant qui va définir toutes les opérations relatives aux *DIDs*, aux *VCs* et à la communication par messages sécurisés. Les standards stables [36] supportés sont actuellement les suivants :
   (a) le Well Known DID Configuration ;
   (b) le JSON Web Encryption ;
   (c) le JSON Web Key ;
   (d) la JSON Web Signature ;
   (e) l'OpenID Connect ;
   (f) les Decentralized Identifiers ;
   (g) le JSON-LD ;
   (h) les Verifiable Credentials.

36. Pour qu'un standard soit considéré comme "stable", celui-ci doit être entièrement supporté et son support doit continuer dans le futur. Les autres standards peuvent être trouvés à cette adresse.

2. **VII Extensions** est le composant qui va permettre d'élargir *VII Core* avec le *OIDC Bridge* qui est une extension permettant aux développeurs d'étendre les capacités des systèmes d'identités actuels avec *MATTR VII*[37]. Il existe également une autre extension, nommée *White-label Mobile Wallet & SDKs*, qui va permettre l'interaction avec les portefeuilles digitaux mobiles.
3. **VII Drivers** est un composant qui contient des intégrations pré-configurés dont le but est de moduler certains paramètres de *MATTR VII*. Il est par exemple possible de changer la méthode de *DID* ou bien la gestion de clés utilisée par *MATTR VII*.
4. **Developer Tools** est un composant qui contient des interfaces dont le but est d'aider le développeur dans son utilisation de *MATTR VII*. Il y a le portefeuille mobile de *MATTR*, des échantillons d'applications qui intéragissent *MATTR VII* et une interface en ligne de commande.

Le développeur, lors de l'utilisation de la solution, va effectuer des requêtes sur l'***API*** de *MATTR VII*. Ces différentes méthodes peuvent être trouvées sur la référence officielle de l'***API***. Pour le développement du client qui va effectuer ces requêtes, n'importe quel langage peut être utilisé, tant que celui-ci supporte ce genre de requêtes. Ainsi, la communication des composants de *MATTR VII* et du client, peut être comparée à celle de la communication entre le ***framework*** et le contrôleur dans *Aries* (MATTR LEARN, s. d.-b). La figure 2.42 représente cette architecture.

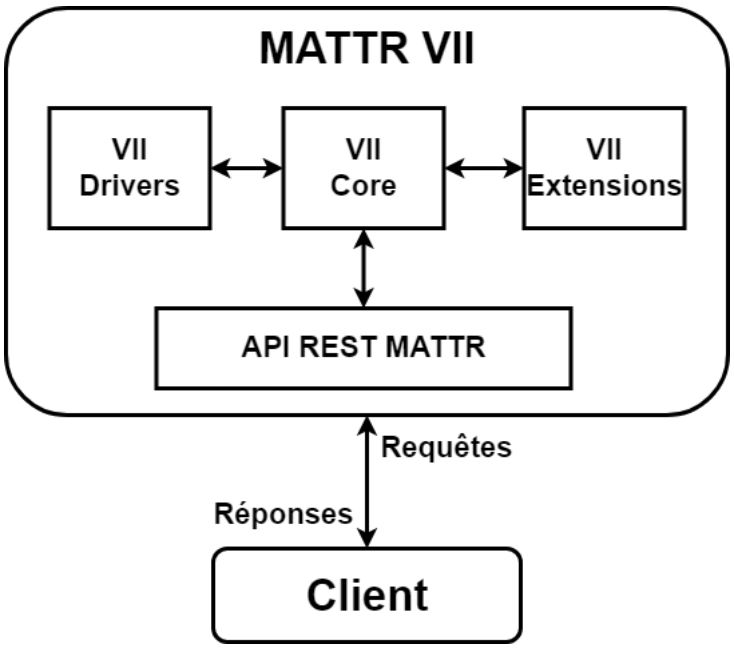


FIGURE 2.42 – *Architecture de MATTR VII*
Source: *de l'auteur*

37. Cette extension permet, par exemple, l'intégration avec le standard OpenID Connect.

Avantages

La force principale de *MATTR VII* est sa facilité d'intégration, puisque le développeur a seulement besoin de contacter l' ***API REST*** à disposition pour l'utiliser. De plus, la documentation et les différentes ressources d'apprentissage à disposition sont de très bonne qualité, ce qui permet un gain de temps pour la prise en main de *MATTR VII*.

Cette solution implémente également de nombreux standards. Le support de l'OpenID Connect, par exemple, est primordial car il permet une intégration avec les systèmes d'identités actuels. La compatibilité avec les systèmes actuels est donc un enjeu majeur puisque le but de ces solutions n'est pas de tout démolir pour ton reconstruire, mais plutôt de renforcer les fondations déjà en place.

Un autre avantage, que nous retrouvons également dans *Aries*, est la liberté de choisir le langage de programmation que le développeur souhaite utiliser pour intégrer la communication avec *MATTR VII* qui permet un gain de temps important.

Inconvénients

L'un des inconvénients de cette solution réside dans le fait qu'elle n'est ni gratuite, ni ***open source*** et qu'il faut s'inscrire pour la tester. De plus, les différents prix ne sont pas disponibles sur leur site internet et il faut donc contacter la société pour connaître les tarifs (MATTR LEARN, s. d.-b).

Un autre aspect problématique de *MATTR VII* est son manque de flexibilité. Effectivement, comme nous l'indique la documentation, il est possible grâce aux *VII Drivers* de changer certains paramètres lors des requêtes envoyées à l'***API***. Cependant, cette ***API*** ne change pas en arrière plan. Un développeur qui souhaiterait construire une solution personnalisée devrait alors bien analyser les fonctionnalités offertes par *MATTR VII* avant d'adopter la solution (BOLTE, 2021).

Un dernier point important, lié à ce manque de flexibilité est que le ***Key Management System*** est fourni par MATTR et il n'est pas possible de le changer. Cela implique que l'entièreté des clés privées des clients sont stockées dans les systèmes de MATTR, ce qui est problématique pour la création d'un ***e-ID*** étatique. Cela signifierait que les clés privées des citoyens seraient sous contrôle d'une société située en Nouvelle-Zélande. Cette caractéristique n'est pas mise en avant sur leur site internet, l'utilisation de leur ***API*** permet cependant de le constater (BOLTE, 2021).

2.3.3 FIDO

FIDO est un consortium de plusieurs grandes sociétés comme *Google*, *Intel*, *Apple* ou encore *Amazon*. *FIDO* est l'abréviation de *"Fast Identification Online"* ce qui en français signifie *"Identification Rapide en Ligne"* (FIDO ALLIANCE, s. d.).

Le but de *FIDO* est d'offrir des standards faciles à utiliser, sécurisés et confidentiels pour l'***authentification multifacteur***. En partenariat avec le ***World Wide Web Consortium***, le consortium a défini toute une liste de spécifications, de protocoles et de standards dont la documentation peut être récupérée à l'adresse suivante. Ces standards sont regroupés, aujourd'hui, sous le nom *FIDO2* (FIDO ALLIANCE, s. d.).

FIDO2 semble être un tournant majeur dans l'authentification en ligne, les sociétés *Google*, *Microsoft* et *Apple* se sont engagées à mettre en place ces standards dans leurs différents services pour offrir une connexion plus rapide, plus simple et plus sécurisée (APPLE, 2022a).

Ces standards sont libres d'accès, ainsi n'importe qui peut les implémenter. Plusieurs types de ressources sont à disposition, pour les personnes souhaitant implémenter cette solution, comme des vidéos, de la documentation technique et des forums de discussions. Il est possible de passer une certification *FIDO*. Notons également qu'en plus des standards techniques, il y a également des lignes directrices sur des sujets comme le design des interfaces utilisateurs (FIDO ALLIANCE, s. d.).

Caractéristiques

Les protocoles utilisent la *cryptographie asymétrique* pour fournir une authentification renforcée. Comme décrit sur la page *"How FIDO Works"* de FIDO ALLIANCE, s. d., le fonctionnement est le suivant : "Lorsqu'un client s'inscrit à un service en ligne, le dispositif [38] de l'utilisateur va créer une paire de clés asymétriques. La clé privée est conservée dans le dispositif et la clé publique est enregistrée sur le service en ligne. L'authentification est ensuite effectuée par le dispositif qui va prouver qu'il possède la clé privée au service en accomplissant un challenge. Les clés privées ne peuvent être utilisées que lorsque l'utilisateur a déverrouillé son dispositif. Le déverrouillage s'effectue par une action sécurisée et simple à réaliser, comme faire glisser un doigt, saisir un code, parler dans un microphone, insérer un dispositif sécurisé ou appuyer sur un bouton. Les protocoles sont conçus pour protéger la vie privée des utilisateurs, ainsi ils ne fournissent pas d'informations pouvant être utilisées par des services en ligne pour suivre un utilisateur [39]. Les informations biométriques, si elles sont utilisées, ne quittent pas l'appareil de l'utilisateur."

Ainsi les phases d'enregistrement et de connexion peuvent être représentées selon les schémas 2.43 et 2.44. Notons que *FIDO2* lie les dispositifs à leurs utilisateurs et aux services sur lesquels ils sont utilisés [40]. Il est important de mentionner que la paire de clés n'a pas besoin d'être générée depuis une tierce partie de confiance, car celles-ci servent uniquement à lier l'utilisateur à son compte sur le service en ligne (GRIMES, 2021).

38. Un dispositif est un appareil électronique compatible avec *FIDO2*. Cela peut être, par exemple, un téléphone mobile, une tablette, un ordinateur, une carte ou encore une clé USB.
39. C'est-à-dire connaître toutes ces activités sur Internet.
40. Un seul dispositif peut servir sur plusieurs services.

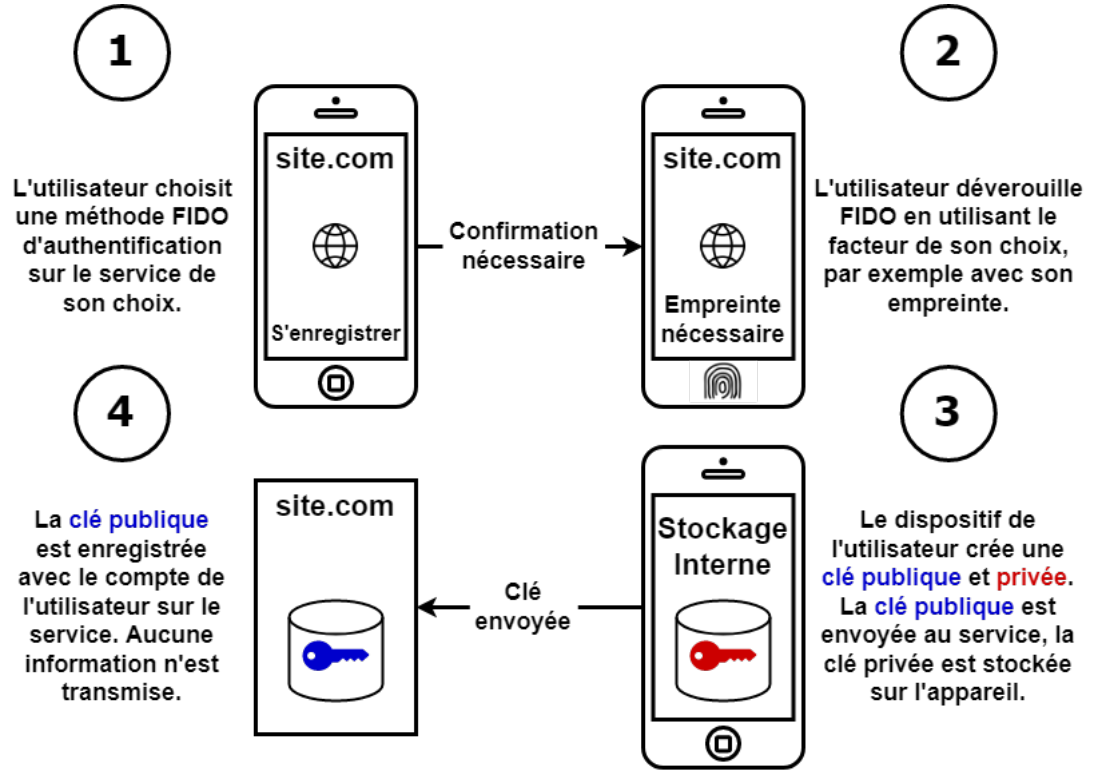


FIGURE 2.43 – *Enregistrement avec FIDO2*
Source: *traduit et modifié par l'auteur depuis "How FIDO Works" sur* FIDO ALLIANCE*, s. d.*

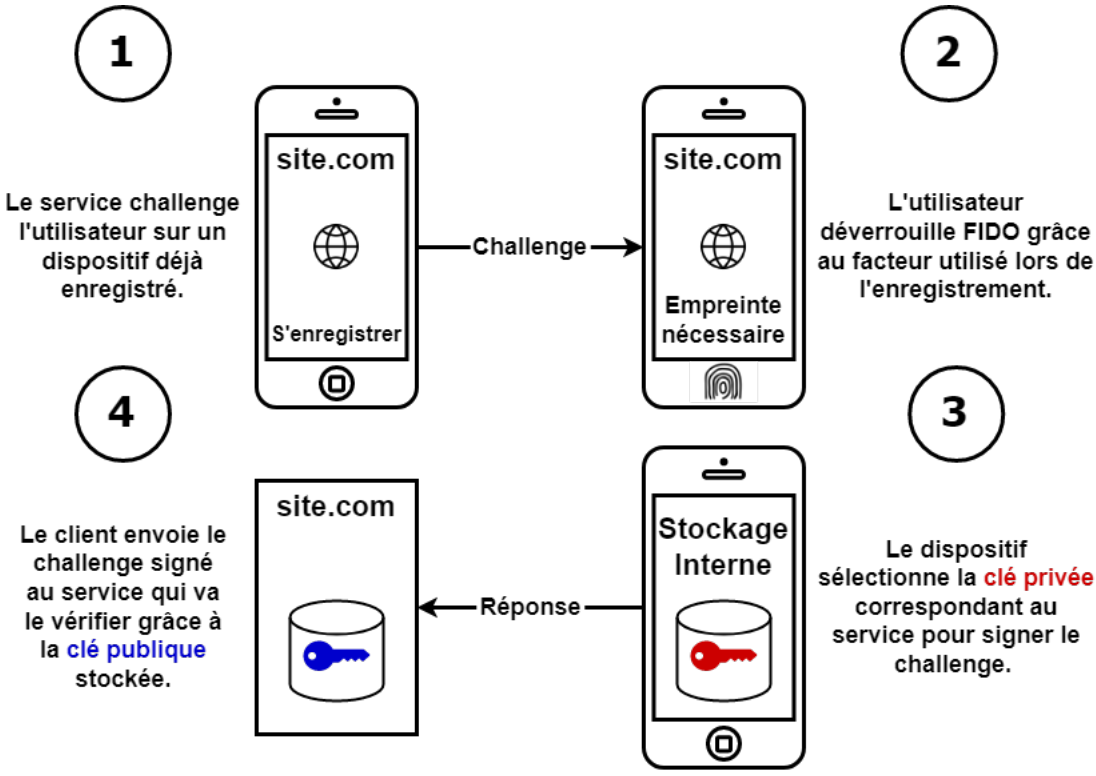


FIGURE 2.44 – *Connexion avec FIDO2*
Source: *traduit et modifié par l'auteur depuis "How FIDO Works" sur* FIDO ALLIANCE*, s. d.*

Avantages

FIDO2 est une technologie mise peu à peu à disposition du public. Lors de l'événement *WWDC22*, la société *Apple* a annoncé que la plupart de ses produits supporteront ce système d'authentification (APPLE, 2022c). C'est un avantage important car plus *FIDO2* se démocratise, plus l'écosystème autour de celui-ci peut se renforcer et de ce fait améliorer également l'expérience utilisateur.

Le support de *FIDO* par des multinationales est également un avantage intéressant, car les capacités techniques, pouvant être mises à disposition du projet, sont importantes. Les ressources également misees à disposition des développeurs sont intéressantes car elles permettent une compréhension et une mise en place rapide de *FIDO2*.

Notons que l'utilisation de cette technologie permet de renforcer la protection contre les attaques informatiques visant les systèmes d'authentification actuels. Par exemple, la réussite de certaines attaques de ***Man-In-The-Middle-Attack*** est quasiment impossible avec *FIDO2* (GRIMES, 2021).

Inconvénients

L'inconvénient principal de cette solution est qu'elle ne règle pas le problème profond du manque d'identité sur Internet. Ainsi, elle ne permet que de se connecter sans un mot de passe et ne règle donc que cet aspect de la problématique. *FIDO2* n'apporte donc pas de solutions dans le cadre d'un échange d'informations sensibles ou dans la conformité des règlements *RGPD* et *LPD*.

De plus, cette technologie reste vulnérable à certains types d'attaques. Par exemple, un dispositif peu sécurisé peut être compromis donnant ainsi l'accès total à une personne malveillante (GRIMES, 2021).

2.3.4 NFT

Un *"Non-Fungible Token (NFT)"*, ou en français *"Jeton Non Fongible"*, est un type de jeton électronique stocké dans un *Distributed Ledger*. Ils ont pour but de représenter une donnée unique, par exemple un objet de collection ou un titre de propriété. Un *NFT* n'a qu'un seul propriétaire, personne ne peut le modifier ou en créer une copie (ETHEREUM, s. d.-b).

La valeur d'un *NFT* est définie selon la loi de l'offre et de la demande. Certaines collections[41] numériques d'oeuvres d'art se sont vendues plus de 50 millions de dollars (HOOD, 2022). C'est une technologie controversée à cause de certaines de ses utilisations ou pour sa consommation énergétique importante en fonction du *DL* sur lequel elle est construite. *Ethereum*, qui supporte les *NFTs*, promet cependant une amélioration de cette consommation dans le futur en passant sur une version 2.0, comme discuté dans le chapitre 2.2.4 (ETHEREUM, s. d.-b).

41. Il est ici question de *collections* car c'est un regroupement de plusieurs *NFT* représentant chacun une oeuvre numérique.

Caractéristiques

Comme nous l'avons dit auparavant, un *NFT* n'a qu'un seul propriétaire. Cette association est possible grâce à l'utilisation d'un identifiant unique et de métadonnées, tous les deux liés au *NFT*. C'est un ***smart contract*** qui va créer un *NFT* en lui attribuant ses propriétés uniques et en définissant la qualité de transfert de celui-ci (ETHEREUM, s. d.-b).

Grâce à son fonctionnement par clé asymétrique, un *NFT* permet à son propriétaire de prouver facilement qu'il en est le propriétaire, car le *NFT* est signé par sa clé publique. Ce système de signature permet également au créateur du *NFT* de le signer de sa clé publique afin de prouver qu'il en est bien le créateur. Un *NFT* est défini par une norme commune, par exemple la norme ERC-721 (ETHEREUM, s. d.-b).

Ces propriétés se montrent particulièrement intéressantes dans le triangle de confiance représenté par la figure 2.22 que nous avons découvert dans un chapitre précédent. Un émetteur peut donc créer un *NFT* pour représenter une identité unique et le transmettre à un détenteur. Ainsi le détenteur peut ensuite facilement prouver au vérificateur qu'il en est bien le propriétaire, et le vérificateur peut à son tour vérifier que l'émetteur, avec qui il a établi un lien de confiance, est bien l'entité émettrice du jeton.

Toutefois, la création d'une identité autogérée grâce aux *NFTs* est limitée. En effet, une autre caractéristique des *NFTs* actuels est le fait de pouvoir être transmissible. Il est possible de vendre ou d'acheter un *NFT*, ce qui dans le cas de la création d'une *SSI* est problématique. Pour bloquer la transmission de ces jetons, il est possible de modifier le comportement des méthodes, par exemple *"safeTransferFrom"*, d'un ***smart contract*** répondant aux normes ERC-721 (ETHEREUM, s. d.-b).

Le 26 janvier 2022, dans un article nommé *"Soulbound"*[42] paru sur son site personnel, Vitalik Buterin, le fondateur d'*Ethereum*, émet l'idée d'un jeton non fongible et non transférable[43]. Il prend en exemple l'achat possible d'un diplôme universitaire ou d'un permis de conduire pour montrer les limites du *NFT* actuel. Un *"Soulbound"* offre alors toutes les caractéristiques d'un *NFT*, tout en permettant que celui-ci ne soit pas échangé (BUTERIN, 2022).

Notons également que ces jetons non échangeables devraient permettre l'utilisation de *Zero-Knowledge Proofs* et des ***smart contracts*** (WESTON, 2022). Des standards sont actuellement en développement pour mettre en place ce genre de technologies, et le nom actuellement proposé est *"Soulbound Badges"*, signifiant *"badges liés à l'âme"*. Les discussions autour de ce nouveau standard peuvent être retrouvées sur le forum de discussions dédié.

42. Le terme peut être traduit en français par *lié à l'âme*.
43. L'appellation non transférable n'est pas tout à fait correcte. Le jeton reste transférable. Cependant, il ne l'est qu'une seule et unique fois, lors du transfert du créateur au propriétaire du jeton.

Avantages

Les *"Soulbound Badges"* représentent une technologie intéressante pour construire sa *SSI*. Ils devraient permettre, comme dit précédemment, l'utilisation de *ZKP* et de ***smart contracts***. De plus, comme nous pouvons le voir dans les standards sur GitHub, il est précisé que les jetons ne seront pas uniquement liés à un être humain mais à un persona [44]. Cet aspect rend l'utilisation de jetons intéressante pour l'*Internet of Things* et s'inscrit tout à fait dans le cadre prévu de la *SSI*. Leurs standards sont également ouverts à tous et chacun est libre de les utiliser.

Inconvénients

Les *NFT*, dans leur forme transférable, ne peuvent pas être utilisés pour la *SSI* mais les *Soulbound Badges* le peuvent. Ils sont cependant en cours de développement, ce qui les rend pour le moment inutilisable. Il est donc difficile de trouver des inconvénients à ceux-ci. Notons cependant que que les *Distributed Ledgers*, sur lesquels ces badges seront construits, doivent répondre aux caractéristiques que nous avons fixé dans le chapitre 2.2.4.

44. N'importe quel type d'entité pouvant être représenté sur Internet.

3 | Défis identifiés par le Département fédéral de justice et police

Comme nous l'avons vu dans les chapitres précédents, l'une des alternatives identifiées pour la mise en place de l'***e-ID*** en Suisse, est la *Self-Sovereign Identity*. Cette solution soulève cependant de nombreuses questions. Dans ce chapitre, nous nous concentrons sur les différents défis posés par le *Département fédéral de justice et police (DFJP)*. Pour chacun de ces défis, nous proposons différentes solutions et exposons également les limites de celles-ci.

3.1 Premier défi

Le premier défi est formulé selon l'énoncé suivant *"Cette approche est relativement récente, certaines questions fondamentales ne sont pas encore résolues de manière concluante et les normes sont encore incomplètes."*.

Dans le chapitre 2.1.8, nous relevions des challenges liés à la **gouvernance** ainsi qu'à la **technologie** qui font écho à ce premier défi posé par le *DFJP*. Dans les défis liés à la gouvernance, nous relevions qu'il manque actuellement un leader dans l'écosystème de la *SSI*, ce qui conduit à des entités indépendantes qui fixent chacune leurs propres règles. Il y a alors un manque de consensus sur des questions fondamentales ainsi que sur les normes proposées. Dans les défis liés à la technologie, nous relevions que l'écosystème autour de la *SSI* est pour l'instant immature, créant ainsi des zones d'ombre sur certaines questions [1].

La question est alors de savoir comment est-il possible de résoudre certaines questions fondamentales et d'avoir des normes complètes.

3.1.1 Solutions

Un élément qui peut apporter une réponse à ces questions est la mise en place d'une architecture commune pour l'interconnexion des différentes technologies de l'écosystème de la *SSI*. C'est ce que propose le fondation *Trust Over IP*, soutenue par la fondation *Linux*, dans son modèle de gouvernance à quatre couches représenté par la figure 3.1 [2]. Cette architecture peut alors amener des lignes directrices pour l'implémentation de la *SSI* à l'échelle d'Internet.

La solution apportée est donc que ces lignes directrices peuvent donner un but commun aux différents acteurs de l'écosystème, ce qui montre à quel point il est important de trouver des consensus.

1. La question de savoir comment faire fonctionner la *SSI*, tout en étant hors-ligne, est un exemple de ces questions qui sont encore sans réponses (PREUKSCHAT et REED, 2021).
2. La figure n'est pas traduite car le but n'est pas d'expliquer son fonctionnement mais de faire découvrir son existence.

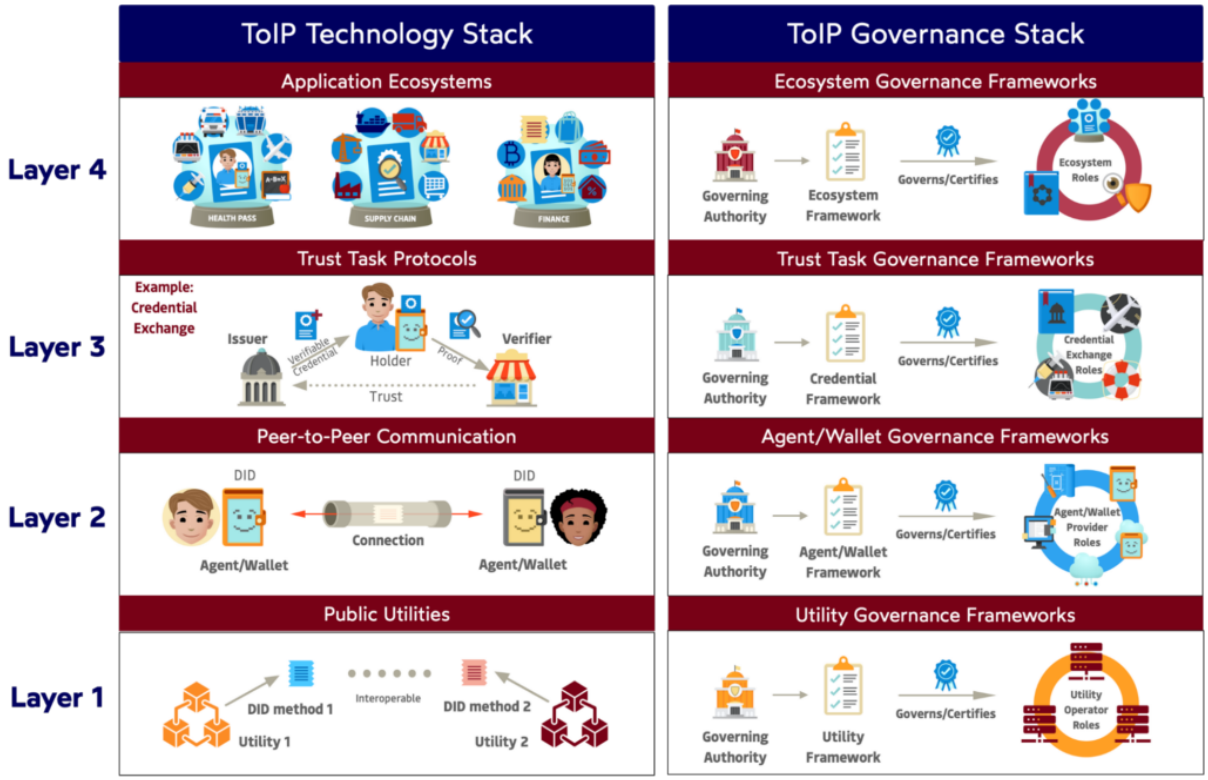


FIGURE 3.1 – *Modèle de gouvernance Trust Over IP*
Source: TRUSTOVERIP*, 2022*

Un autre élément qui peut répondre à ces problématiques est l'apparition d'un leader dans l'écosystème de la *SSI*. Il permettrait de *"donner l'exemple"* aux autres acteurs, que ce soit d'un point de vue technique - sur sa façon d'implémenter la *SSI* - ou d'un point de vue philosophique - sur sa façon de *"penser"* la *SSI*.

La Confédération suisse pourrait devenir, grâce à son idée d'implémenter une ***e-ID*** à l'échelle du pays, ce leader recherché. En effet, la Suisse serait le premier pays à mettre en place un tel système avec l'aide de l'identité autogérée, ce qui lui permettrait d'utiliser les technologies de son choix. D'autres pays pourraient également suivre la Suisse dans son exemple. Les technologies utilisées deviendraient alors des références dans le domaine, obligeant les différents acteurs à s'y adapter. Si un acteur offre une solution complète, adéquate, facile à mettre en place et répondant aux critères de confidentialités et sécurités nécessaires, celui-ci peut alors également devenir le leader du secteur. Ce leader définirait ainsi les bases de ce qu'est la *Self-Sovereign Identity*, au même titre qu'*Apple* a établi les bases de qu'est aujourd'hui le *smartphone*.

Ainsi la combinaison de normes communes et de l'apparition d'un leader pourrait permettre de répondre à des questions fondamentales et d'avoir des normes complètes, en obligeant les différentes entités à travailler dans un but commun, ou en suivant un même exemple.

3.1.2 Limites

Le problème avec ces solutions est qu'elles posent elles-mêmes des nouveaux défis. En effet, la problématique devient alors de savoir comment faire collaborer ces différents acteurs dans un but commun. Qui proposera des normes communes ? Est-ce que les différentes entités vont accepter de suivre ces normes ? Même avec l'apparition d'un leader, est-ce que les acteurs vont choisir de le prendre comme exemple ou au contraire proposer leurs propres visions ?

Il est difficile de donner des réponses à toutes ces questions pour le moment, car seul le temps peut nous dire comment l'écosystème va évoluer. Nous pouvons essayer d'imaginer le futur, mais nous ne pourrons jamais être certains sur la tournure que vont prendre les événements.

Comme nous l'avons dit, les solutions les plus plausibles sont, pour le moment, l'apparition d'un leader et la mise en place de normes complètes grâce à des buts communs. Cependant, il est impossible d'affirmer que ces solutions auront l'effet escompté. Ainsi, la meilleure solution pour répondre à ce défi est de ne pas se précipiter et de laisser le temps faire son travail.

3.2 Deuxième défi

Le deuxième défi est formulé selon l'énoncé suivant *"L'opinion publique doit d'abord être sensibilisée au potentiel de cette approche globale, qui diffère d'un **Login**."*.

Comme nous l'avons vu tout au long de ce travail, la *SSI* représente une nouvelle approche pour l'authentification en ligne, car c'est un nouveau modèle d'identité 2.1.2. Cette technologie se veut être révolutionnaire dans la façon dont les utilisateurs vivent sur Internet. Selon les auteurs du livre PREUKSCHAT et REED, 2021, ce changement est plus qu'un changement technologique sur Internet, c'est une transition profonde de l'infrastructure et de la dynamique d'Internet même. Ils comparent ce changement au passage des voyages en train à ceux en voiture.

Mais comme toute révolution technologique, celle-ci aura droit à ses fidèles et à ses détracteurs. Il est important, surout dans l'époque où nous vivons où l'information est rapide, d'éviter la désinformation autour du sujet. Celle-ci ruinerait potentiellement des heures de travail investies pour la mise en place d'un système qui se veut pour la protection et le confort de ses usagers.

Nous pouvons alors nous demander comment serait-il possible de sensibiliser l'opinion publique sur la *SSI*, tout en évitant la désinformation sur le sujet.

3.2.1 Solutions

Il est souvent constaté que les avis des Suisses divergent en fonction de la région qu'ils habitent. Le terme *"Röstigraben"* désigne même ce contraste entre la population de Suisse romande et celle de Suisse alémanique. Un élément important est d'éviter d'imposer cette technologie à la population. C'est également le chemin que semble prendre le Conseil fédéral car l'***e-ID*** devrait être facultatif (BARBEY, 2022).

Ne pas imposer l'identité autogérée peut déjà représenter un impact positif sur l'avis de la population. En effet, l'état ne nous impose rien mais nous propose plutôt un nouvel outil. De plus, le fait qu'il soit facultatif permet également d'éviter les tensions que la Suisse a connue lors des débats animés autour de la pandémie de Covid-19 [3].

Une autre solution pour sensibiliser les gens à cette nouvelle approche est d'utiliser les moyens de communication actuels pour transmettre les informations. Il peut, par exemple, être intéressant de mettre en place des vidéos de vulgarisation pour expliquer différentes notions relatives à la *SSI* [4]. Pour ce faire, il est possible de publier ces vidéos sur la plateforme *Youtube* ou alors sous forme de publicité [5]. Une campagne d'information semble alors intéressante pour sensibiliser l'opinion publique.

Notons également que le journalisme joue un rôle important sur l'opinion publique. Des émissions diffusées sur la *Radio télévision suisse*, comme *T.T.C. (Toutes taxes comprises)*, offrent un regard critique aux téléspectateurs, ce qui est capital pour combattre la désinformation.

Finalement, toutes ces solutions ont pour but de montrer l'important de faire comprendre la nécessité d'un nouveau système, comme celui de la *SSI*, au peuple et aux entreprises. Comme nous l'avons vu dans le chapitre 2.1.1, l'urgence autour de la situation du ***cybercrime*** ne fait qu'accentuer cette nécessité. Puisque la population n'est peut-être pas au courant de l'impact du ***cybercrime***, il est important de présenter à cette population les bénéfices d'un nouveau système afin de la sensibiliser à ce danger.

3.2.2 Limites

La limite de cette solution est que ces campagnes doivent servir d'informations pour sensibiliser les gens et éviter la désinformation. Elles ne doivent pas devenir des campagnes de propagande. Comme nous l'avons vu tout au long de ce document, la *SSI* a des avantages mais également des inconvénients. Il est donc important de présenter tous ces aspects de la *SSI*, tout en évitant que la population ne se sente envahie par ces campagnes de sensibilisation, les individus pouvant alors se désintéresser ou même être incommodés par la situation.

Un dernier point important est la question du financement. Les coûts des campagnes ne doivent pas être exceptionnels car cela pourrait conduire vers des critiques de la population.

Finalement, les personnes prenant la parole sur de telles technologies doivent avoir un minimum de connaissances techniques sur celles-ci pour éviter de promouvoir un composant incompris.

3. La pandémie du Covid-19 est une pandémie mondiale provoquée par le coronavirus SARS-CoV-2.
4. La vidéo suivante explique ce qu'est l'identité électronique étatique.
5. Comme c'est le cas lors de votations sur des sujets techniques.

3.3 Troisième défi

Le troisième défi est formulé selon l'énoncé suivant *"La responsabilité de la gestion des données vérifiées est pleinement confiée à l'utilisateur, ce qui rend toute aide de l'émetteur quasiment impossible."*.

Dans le modèle d'identité traditionnel et fédéré, les données de l'utilisateur ne sont pas conservées par celui-ci mais par l'organisation à laquelle il se connecte. Notre date d'anniversaire, par exemple, indiquée sur les réseaux sociaux est actuellement stockée dans les bases de données de ces services.

Comme nous l'avons vu tout au long de ce document, avec la *SSI*, le but est que ces organisations ne stockent plus ces informations sur l'utilisateur et que celles-ci restent sa propriété. De ce fait, comme pour les cartes dans un portefeuille physique, les *"cartes"* dans un portefeuille virtuel sont sous la responsabilité de son détenteur. L'utilisateur est donc responsable en cas de perte ou de vol, par exemple.

C'est une subtilité dont la compréhension est capitale, car des fonctionnalités comme *"Mot de passe oublié"* sur les services web ne seront plus possibles avec l'introduction de la *SSI*. Il est également important de noter que tous les processus d'obtention de clés cryptographies ou de justificatifs sont également de la responsabilité des utilisateurs.

Nous pouvons alors nous demander comment aider, au mieux, l'utilisateur dans cette gestion de ses propres données.

3.3.1 Solutions

Cette question est étroitement liée au premier défi posé par le *DFJP*. Il fait partie de ces problématiques dont les résolutions dépendent des différentes solutions disponibles et de la compréhension que les organisations ont du terme *SSI*. Cela montre encore une fois l'importance d'atteindre un consensus sur certaines normes et définitions dans le domaine.

Une solution identifiée par les auteurs du livre PREUKSCHAT et REED, 2021 est de mettre en place une assurance qui va assurer le détenteur contre le vol ou la perte de son portefeuille. Cette solution fait écho au fait que le détenteur du portefeuille n'est pas forcément un individu mais peut également être une entité non physique, comme une organisation par exemple. Le sentiment de protection de ces institutions, et des personnes physiques aussi, peut alors être amené grâce à ce système d'assurances.

Finalement, pour aider les utilisateurs dans leur obtention de clés asymétriques ou de justificatifs, il sera nécessaire de mettre une attention particulière sur l'expérience utilisateur des solutions proposées. Nous pouvons imaginer des guides disponibles dans les portefeuilles digitaux, une aide vidéo pour effectuer certaines étapes ou encore de directement avoir une aide physique, en se rendant dans une administration par exemple.

3.3.2 Limites

Il faut bien comprendre que ce n'est pas la question d'amener de l'aide qui pose problème, mais de réussir à implémenter cette aide tout en gardant les spécificités de la *SSI*. Par exemple, une aide externe ne doit pas avoir accès aux données de l'utilisateur, car cela représenterait une brèche de confidentialité et de sécurité.

L'assurance ne nous permet pas de récupérer nos données, celles-ci sont perdues. Elle nous permet uniquement de nous protéger contre et lors d'un vol ou d'une perte. De plus, l'assurance serait payante ce qui peut représenter, pour certains utilisateurs, un frein pour sa contraction.

Pour conclure, les solutions pour accompagner les utilisateurs lors de l'obtention de justificatifs ou de clés nécessitent également un investissement financier de la part des fournisseurs de solutions, ou des autorités administratives compétentes pour les mettre en place. Cet investissement peut représenter un frein à l'expérience utilisateur car certaines organisations ne voudront ou ne pourront pas financer de telles fonctionnalités et services.

3.4 Quatrième défi

Le quatrième et dernier défi est formulé selon l'énoncé suivant *"Les possibilités d'évaluation forensique sont réduites, car le système bénéfice d'une bonne protection décentralisée et cryptographique. En cas d'utilisation abusive de l'**e-ID** ou d'autres preuves, il peut donc être difficile de prouver qu'on "n'était pas" la personne en question."*.

Nous avons vu tout au long de ce document que les technologies utilisées pour la mise en place de la *SSI* offrent une couche d'identité sur Internet. L'utilisation de registre distribué ou de mécanismes de protection cryptographique rendent cette identité très difficile à dérober.

Cela n'est cependant pas impossible. Si le portefeuille numérique est stocké sur un appareil non sécurisé ou que lui-même ne l'est pas, il suffit que quelqu'un dérobe l'appareil en question pour pouvoir alors utiliser les justificatifs stockés dans le portefeuille numérique. Ainsi si un malfaiteur utilise la carte de crédit d'un utilisateur, il est difficile pour le vrai détenteur de prouver que ce n'est pas lui.

La question est alors de savoir comment est-il possible d'éviter ce genre de situation et de renforcer la sécurité de l'identité de l'utilisateur.

3.4.1 Solutions

Dans le rapport HARDMAN, 2019, la fondation *Sovrin* offre une idée de solution à mettre en place en cas de vol de son téléphone. La première étape de cette solution est de révoquer les droits d'utilisation du téléphone dérobé. Cela peut être fait par l'utilisateur en utilisant un

autre dispositif sous son contrôle [6]. La deuxième étape de la solution est la révocation des clés utilisées, grâce au même appareil que dans la première étape. Cette solution propose donc une réponse en cas de vol d'un dispositif.

Dans le même rapport, il est également proposé plusieurs techniques pour renforcer la protection du portefeuille numérique, comme l'utilisation de l'***authentification multifacteur***, ou encore le renforcement des règles sur l'utilisation de son portefeuille, comme par exemple interdire l'utilisation d'un justificatif pendant certaines périodes (HARDMAN, 2019).

Une autre technologie qui peut être mis en place pour renforcer l'identification de l'utilisateur, lorsqu'il utilise son portefeuille, est le *"Perceptual Hasing"* ou en français le *hachage perceptuel*. C'est une fonction de ***hachage*** qui est utilisée pour le contenu multimédia, par exemple des images. Elle est particulièrement intéressante pour la comparaison de deux contenus multimédias car contrairement aux méthodes de ***hachage*** traditionnels, cette fonction n'est pas sensible aux petits changements lors de l'entrée des données mais est sensible aux différences dans la perspective des objets comparés (DU, HO et CONG, 2020).

Cette fonction se montre alors particulièrement intéressante pour le renforcement de la reconnaissance faciale ou digitale d'un utilisateur. Elle est par exemple utile lors d'attaques par inversion de modèle (PRAKASH et al., 2020) [7].

Nous pouvons donc imaginer qu'à chaque fois qu'un utilisateur souhaite utiliser un justificatif, une photo soit prise de son visage pour créer un ***digest*** et l'inscrire dans une transaction. Il est ensuite possible de vérifier l'historique de ces ***digests*** pour vérifier que c'est le vrai détenteur qui utilise le portefeuille numérique.

3.4.2 Limites

Même s'il est possible de renforcer de plus en plus techniquement les différents systèmes, ceux-ci restent vulnérables aux attaques de ***social engineering***. Seule la prévention contre ce type d'attaques peut permettre de réduire les risques liés à celles-ci.

Notons également que l'***authentification multifacteur*** peut être attaquée. Elle représente donc une couche de sécurité supplémentaire mais pas infaillible. Des ouvrages comme GRIMES, 2021 sont d'ailleurs dédiés à la présentation de différentes attaques contre ce type de protection. La solution proposée par la fondation *Sovrin* ouvre également la question de savoir comment est-il possible de lier plusieurs appareils à un même compte. De plus, la révocation d'un appareil ou de clés ou l'enregistrement de plusieurs dispositifs ne sont pas implémentés par tous les fournisseurs, ce qui montre l'importance d'avoir un consensus sur la question.

Finalement, pour ce qui est du *"Perceptual Hasing"*, celui-ci ne semble pas non plus être invincible. L'article WENG et PRENEEL, 2007 démontre certaines faiblesses de cette fonction.

6. Cette étape porte le nom de *"device revocation"* ou en français *"révocation de dispositif"*.
7. Cette attaque consisterait à reconstruire une image à partir d'un modèle de *Machine Learning*. Nous n'entrerons pas dans les détails de ces différentes notions car elles dépassent la portée de ce document.

4 | Proof of concept

Dans le chapitre suivant, nous nous concentrons sur les différentes preuves de concept réalisées pendant ce travail. Le terme anglais utilisé pour la preuve de concept est le Proof of Concept (PoC). Nous utilisons désormais cet acronyme pour la suite de ce document. Les *PoCs* que nous présentons permettent de montrer l'utilisation de la solution *Hyperledger Aries* pour la mise en place d'une *SSI*.

Ces *PoCs* sont la reproduction de démonstrations, appelées *labs*, trouvables sur le lien suivant. Ces exemples sont également disponibles en suivant le cours *Becoming a Hyperledger Aries Developer*. Ainsi, pour chaque *PoC*, nous présentons leur mise en place et synthétisons celle-ci dans un résultat.

4.1 Création d'une connexion avec Swagger

Dans cette démonstration, nous reproduisons le *lab* suivant sur une machine ***Wsl Ubuntu*** s'exécutant sur ***Windows*** 11. Pour cette exemple, nous utilisons l'*Aries Cloud Agent Python* ainsi que *Docker* et *Git*.

4.1.1 Prérequis

Pour commencer, nous clonons l'archive du *Cloud Agent Python*. Cette archive dispose d'un dossier nommé *"demo"* dans lequel des agents préconfigurés [1], des marches à suivre et divers exemples de l'utilisation du ***framework*** se trouvent. Pour cette démonstration, nous lançons l'un de ces agents en utilisant un *Distributed Ledger* nommé *"bcovrin"*.

```
valonrexhepi@WINDOWSDESKTOP:~$ git clone
↪  https://github.com/hyperledger/aries-cloudagent-python
valonrexhepi@WINDOWSDESKTOP:~$ cd aries-cloudagent-python/demo/
valonrexhepi@WINDOWSDESKTOP:~/aries-cloudagent-python/demo$
↪  LEDGER_URL=http://dev.greenlight.bcovrin.vonx.io ./run_demo faber
Preparing agent image...
...
Faber      |
Faber      | ::::::::::::::::::::::::::::::::::::::::::::::
Faber      | :: faber.agent                             ::
Faber      | ::                                         ::
Faber      | :: Inbound Transports:                     ::
Faber      | ::                                         ::
Faber      | ::   - http://0.0.0.0:8020                 ::
Faber      | ::                                         ::
Faber      | :: Outbound Transports:                    ::
Faber      | ::   - http                                ::
```

1. Dans ce contexte, la communauté *Hyperledger* utilise le terme *"agent"* pour désigner une instance du ***framework*** s'exécutant sur un conteneur. Cette instance permet à un utilisateur d'effectuer différentes opérations, comme l'échange de *DIDs*, à travers des interfaces préconfigurées.

```
Faber      | ::   - https                                  ::
Faber      | ::                                            ::
Faber      | :: Public DID Information:                    ::
Faber      | ::   - DID: ELTPPKmwqivorTwwSPSvX7            ::
Faber      | ::                                            ::
Faber      | :: Administration API:                        ::
Faber      | ::                                            ::
Faber      | ::   - http://0.0.0.0:8021                    ::
Faber      | ::                                            ::
Faber      | ::                                  ver: 0.7.4 ::
Faber      | ::::::::::::::::::::::::::::::::::::::::::::::::
Faber      |
Faber      | Listening...
Faber      |
Startup duration: 5.02s
Admin URL is at: http://192.168.65.3:8021
Endpoint URL is at: http://192.168.65.3:8020
#3/4 Create a new schema/cred def on the ledger
Schema:
  {
    "sent": {
      "schema_id": "ELTPPKmwqivorTwwSPSvX7:2:degree schema:57.54.35",
      "schema": {
        "ver": "1.0",
        "id": "ELTPPKmwqivorTwwSPSvX7:2:degree schema:57.54.35",
        "name": "degree schema",
        "version": "57.54.35",
        "attrNames": [
          "birthdate_dateint",
          "date",
          "degree",
          "timestamp",
          "name"
        ],
        "seqNo": 142670
      }
    },
    "schema_id": "ELTPPKmwqivorTwwSPSvX7:2:degree schema:57.54.35",
    "schema": {
      "ver": "1.0",
      "id": "ELTPPKmwqivorTwwSPSvX7:2:degree schema:57.54.35",
      "name": "degree schema",
      "version": "57.54.35",
      "attrNames": [
        "birthdate_dateint",
        "date",
        "degree",
        "timestamp",
        "name"
      ],
      "seqNo": 142670
    }
  }
Schema ID: ELTPPKmwqivorTwwSPSvX7:2:degree schema:57.54.35
Cred def ID: ELTPPKmwqivorTwwSPSvX7:3:CL:142670:faber.agent.degree_schema
Publish schema/cred def duration: 8.33s
...
Waiting for connection...
```

Arrêtons-nous un instant pour comprendre ce qui vient de se passer. La commande *"./run_demo"*, à la ligne 3, exécute un script nommé *"run_demo"*[2]. Cette exécution démarre un conteneur *Docker*[3] qui lance un agent préconfiguré.

Ce script permet de lancer plusieurs agents différents. Le paramètre à la fin de la commande précise celui que nous souhaitons démarrer. Lorsque nous passons la valeur *"faber"*, cela va indiquer au script de lancer l'agent nommé Faber[4]. Comme nous l'avons dit, ces agents sont préconfigurés, ce qui signifie qu'ils ont des paramètres prédéfinis, comme leur port d'exécution par exemple. Nous pouvons voir les autres agents disponibles et leurs paramètres dans le code du script.

```
1  if [ "$AGENT" = "faber" ]; then
2    AGENT_MODULE="faber"
3    AGENT_PORT=8020
4    AGENT_PORT_RANGE=8020-8029
5  elif [ "$AGENT" = "alice" ]; then
6    AGENT_MODULE="alice"
7    AGENT_PORT=8030
8    AGENT_PORT_RANGE=8030-8039
9  elif [ "$AGENT" = "acme" ]; then
10   AGENT_MODULE="acme"
11   AGENT_PORT=8040
12   AGENT_PORT_RANGE=8040-8049
13 elif [ "$AGENT" = "performance" ]; then
14   AGENT_MODULE="performance"
15   AGENT_PORT=8050
16   AGENT_PORT_RANGE=8050-8069
17 else
18   echo "Please specify which agent you want to run. Choose from 'faber', 'alice', 'acme',
     ↪  or 'performance'."
19   exit 1
20 fi
```

Toujours à la ligne 3, l'instruction *"LEDGER_URL=http ://dev.greenlight. bcovrin.vonx.io"* définit, à une variable nommée *"LEDGER_URL"*, un lien *URL* vers le registre distribué utilisé. Lors du lancement du conteneur, cette variable est récupérée dans le script pour permettre la liaison avec le *DL* et son bloc *Genesis*.

```
1  if ! [ -z "$LEDGER_URL" ]; then
2    GENESIS_URL="${LEDGER_URL}/genesis"
3    DOCKER_ENV="${DOCKER_ENV} -e LEDGER_URL=${LEDGER_URL}"
4  fi
```

2. Ce script est écrit en langage Bash et est consultable dans le dossier *"demo"* de l'archive clonnée ou depuis *GitHub*
3. L'explication de ce qu'est un conteneur *Docker* dépasse la portée de ce travail, plus d'informations sur le sujet peuvent être trouvées au moyen de la documentation.
4. Faber est le nom d'un individu que nous incarnons pour la démonstration présentée.

Les *DID publics*, la définition des schémas[5] et des justificatifs, ainsi que les révocations sont stockés dans le *DL*. Dans les informations reçues, lors de l'exécution de la commande *"./run_demo"*, nous voyons les éléments suivants.

```
...
Faber      | :: Public DID Information:                  ::
Faber      | ::                                          ::
Faber      | ::   - DID: ELTPPKmwqivorTwwSPSvX7          ::
...
Schema:
  {
    "sent": {
      "schema_id": "ELTPPKmwqivorTwwSPSvX7:2:degree schema:57.54.35",
      "schema": {
        "ver": "1.0",
        "id": "ELTPPKmwqivorTwwSPSvX7:2:degree schema:57.54.35",
        "name": "degree schema",
        "version": "57.54.35",
        "attrNames": [
          "birthdate_dateint",
          "date",
          "degree",
          "timestamp",
          "name"
        ],
        "seqNo": 142670
      }
    },
    "schema_id": "ELTPPKmwqivorTwwSPSvX7:2:degree schema:57.54.35",
    "schema": {
      "ver": "1.0",
      "id": "ELTPPKmwqivorTwwSPSvX7:2:degree schema:57.54.35",
      "name": "degree schema",
      "version": "57.54.35",
      "attrNames": [
        "birthdate_dateint",
        "date",
        "degree",
        "timestamp",
        "name"
      ],
      "seqNo": 142670
    }
  }
 ...
```

Ces lignes nous indiquent que le *DID* public, ainsi que la définition d'un schéma et de son justificatif, ont été inscrits sur le registre distribué. Nous pouvons vérifier cette information en nous rendant sur l'historique de toutes les transactions du *DL* et en filtrant les données par la valeur du *DID "ELTPPKmwqivorTwwSPSvX7"*. Le lien suivant permet d'afficher les transactions réalisées par l'agent Faber.

5. L'écriture d'un schéma permet de s'assurer qu'un *Verifiable Credentials* respecte une structure définie. Plus d'informations sur les schémas sont trouvables sur le lien suivant.

La figure 4.1 montre l'inscription de la définition du schéma dont les attributs sont : un diplôme, une date de naissance, un horodatage, un nom et une date.

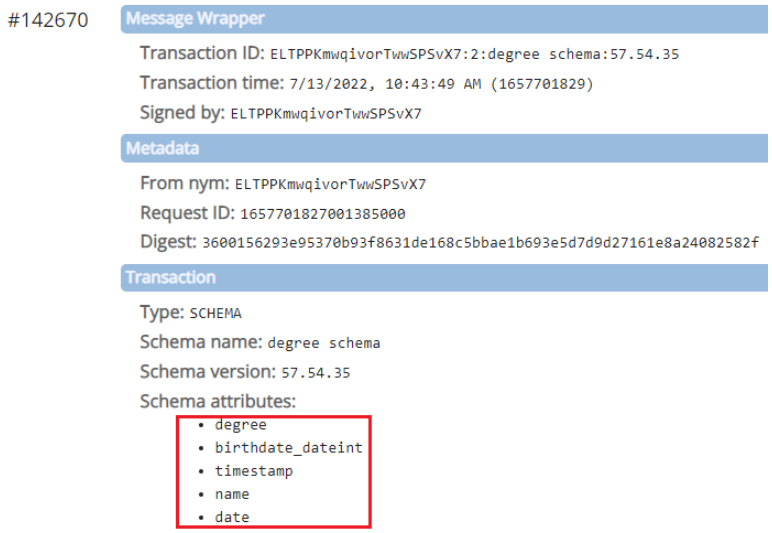


FIGURE 4.1 – *Inscription d'un schéma dans un registre distribué*
Source: *de l'auteur*

Toutes les autres données, comme les invitations ou les connexions, sont inscrites localement. Cette inscription se fait en mémoire ou sur une base de données. Nous découvrons cette spécificité en lisant les différents fichiers constituant le ***framework***.

```
valonrexhepi@WINDOWSDESKTOP:~/aries-cloudagent-python/aries_cloudagent/storage$ ls
askar.py  base.py  error.py  indy.py  __init__.py  in_memory.py  record.py  tests  vc_holder
```

Selon nos recherches, le fichier *"askar.py"* définit le fonctionnement du stockage dans une base de données locale, nommée *Askar*, qui est spécialement conçue pour l'utilisation avec *Aries*. Le fichier *"in_memory.py"* définit la gestion du stockage en mémoire locale. Le fichier *"indy.py"* définit les opérations de stockage sur un registre distribué.

Toutes les informations récupérées par le script lui permettent de lancer, dans sa dernière instruction, le conteneur *Docker*.

```
$DOCKER run --name $AGENT --rm -it ${DOCKER_OPTS} \
  --network=${DOCKER_NET} \
  -p 0.0.0.0:$AGENT_PORT_RANGE:$AGENT_PORT_RANGE \
  ${DOCKER_VOL} \
  $DOCKER_ENV \
  faber-alice-demo $AGENT_MODULE --port $AGENT_PORT $ARGS
```

Nous pouvons reconnaître *"$AGENT"* qui contient le nom *"faber"*, *"$DOCKER_ENV"* qui contient l'*URL* du *Distributed Ledger* et *"$AGENT_PORT"* qui indique le port de l'agent.

Après son lancement, le conteneur est actif.

```
valonrexhepi@WINDOWSDESKTOP:~$ docker container ls
CONTAINER ID   IMAGE              COMMAND                  PORTS
↪    NAMES
a3bb7af33036   faber-alice-demo   "bash -c 'demo/ngrok... "
↪  0.0.0.0:8020-8029->8020-8029/tcp   faber
```

Ce conteneur contient tous les éléments nécessaires à l'exécution de l'instance du ***framework*** *Cloud Agent Python*. Nous pouvons le vérifier en ouvrant un terminal sur le conteneur et en affichant son contenu.

```
valonrexhepi@WINDOWSDESKTOP:~$ docker container exec -it faber /bin/bash
indy@a3bb7af33036:~$ ls -l
total 72
-rw-r--r--  1 root root 10521 Jul 13 06:46 README.md
drwxr-xr-x 25 root root  4096 Jul 13 07:48 aries_cloudagent
drwxr-xr-x  2 indy indy  4096 Jul 13 07:49 aries_cloudagent.egg-info
drwxr-xr-x  1 indy indy  4096 Jul 13 07:48 bin
drwxrwxr-x  1 indy indy  4096 Jul 13 07:49 demo
drwxrwxr-x  1 root root  4096 Feb 24  2021 ledger
drwxrwxr-x  1 root root  4096 Feb 24  2021 log
drwxrwxr-x  2 indy indy  4096 Jul 13 07:49 logs
-rw-r--r--  1 root root    51 Jul 13 06:46 requirements.askar.txt
-rw-r--r--  1 root root    26 Jul 13 06:46 requirements.bbs.txt
-rw-r--r--  1 root root   308 Jul 13 06:46 requirements.dev.txt
-rw-r--r--  1 root root    23 Jul 13 06:46 requirements.indy.txt
-rw-r--r--  1 root root   482 Jul 13 06:46 requirements.txt
drwxr-xr-x  2 root root  4096 Jul 13 07:48 scripts
-rw-r--r--  1 root root  1684 Jul 13 06:46 setup.py
```

Le dossier *"aries_cloudagent"* contient, par exemple, toutes les définitions des protocoles utilisés.

Lorsque nous affichons le conteneur actif, nous constatons également que la commande *"bash -c demo/ngrok-wait.sh"* s'est exécutée automatiquement lors du démarrage. Le contenu du fichier *"ngrok-wait.sh"* contient le code suivant.

```
python -m demo.runners.$AGENT_NAME $@
```

Cette ligne va automatiquement exécuter le contenu d'un fichier Python portant le nom de l'agent *"$AGENT_NAME.py"* et se trouvant dans le dossier *"demo/runners/"*. Pour l'agent Faber, c'est donc le fichier *"faber.py"*. Celui-ci représente le contrôleur de l'agent de Faber. Voici la liste des autres contrôleurs disponibles. Nous discutons du fonctionnement de ceux-ci dans le prochain *PoC*. Notons, pour le moment, que c'est le contrôleur qui va demander l'écriture du *DID* et des définitions sur le registre distribué.

```
indy@a3bb7af33036:~/demo/runners$ ls
__init__.py  acme.py  agent_container.py  alice.py  faber.py  performance.py  support
```

Le lancement d'un agent génère également une interface ***Swagger***. Cette interface nous permet de visualiser toutes les méthodes exposées par le ***framework***. Il est ainsi possible de voir la définition de ces méthodes, des exemples de leur utilisation mais également de les exécuter. C'est cette interface que nous utilisons dans cette démonstration. Son contenu est généré depuis le fichier *"openapi.json"* du dossier *"open-api"* de l'archive.

```
valonrexhepi@WINDOWSDESKTOP:~/aries-cloudagent-python/open-api$ ls
openapi.json  openAPIJSON.config
```

Il y a également une interface en ligne de commandes qui est générée, nous l'utilisons dans le prochain *PoC*. Pour des raisons de simplicité, elle ne sera donc pas plus mentionnée pour le moment. Pendant cette démonstration, nous effectuons différentes requêtes ***REST*** au travers de l'interface ***Swagger***. Des protocoles vont ensuite permettre la liaison entre ces requêtes et le ***framework***. Ces différents protocoles, ainsi que leur définition, se trouvent dans le dossier *"protocols"* du ***framework***.

```
indy@a3bb7af33036:~/aries_cloudagent/protocols$ ls -l
total 76
-rw-r--r-- 1 root root 3397 Jul 13 06:46 README.md
-rw-r--r-- 1 root root    0 Jul 13 06:46 __init__.py
drwxr-xr-x 3 root root 4096 Jul 13 07:48 actionmenu
drwxr-xr-x 3 root root 4096 Jul 13 07:48 basicmessage
drwxr-xr-x 3 root root 4096 Jul 13 07:48 connections
drwxr-xr-x 3 root root 4096 Jul 13 07:48 coordinate_mediation
-rw-r--r-- 1 root root 1766 Jul 13 06:46 didcomm_prefix.py
drwxr-xr-x 3 root root 4096 Jul 13 07:48 didexchange
drwxr-xr-x 4 root root 4096 Jul 13 07:48 discovery
drwxr-xr-x 3 root root 4096 Jul 13 07:48 endorse_transaction
drwxr-xr-x 3 root root 4096 Jul 13 07:48 introduction
drwxr-xr-x 4 root root 4096 Jul 13 07:48 issue_credential
drwxr-xr-x 3 root root 4096 Jul 13 07:48 notification
drwxr-xr-x 3 root root 4096 Jul 13 07:48 out_of_band
drwxr-xr-x 6 root root 4096 Jul 13 07:48 present_proof
drwxr-xr-x 3 root root 4096 Jul 13 07:48 problem_report
drwxr-xr-x 4 root root 4096 Jul 13 07:48 revocation_notification
drwxr-xr-x 3 root root 4096 Jul 13 07:48 routing
drwxr-xr-x 2 root root 4096 Jul 13 07:48 tests
drwxr-xr-x 3 root root 4096 Jul 13 07:48 trustping
```

Par exemple, le dossier *"connections"* définit un protocole du même nom et contient la définition des différentes méthodes pour la gestion des connexions.

Grâce aux différents éléments que nous venons de rencontrer, nous pouvons renforcer nos connaissances sur le fonctionnement du ***framework*** *Cloud Agent Python*. Nous disposons également de toutes les ressources nécessaires pour le bon fonctionnement de la suite de ce *PoC*.

4.1.2 Architecture de départ

Avec les informations définies dans la section précédente, nous pouvons définir que le démarrage de l'agent de Faber génère l'architecture représentée par la figure 4.2.

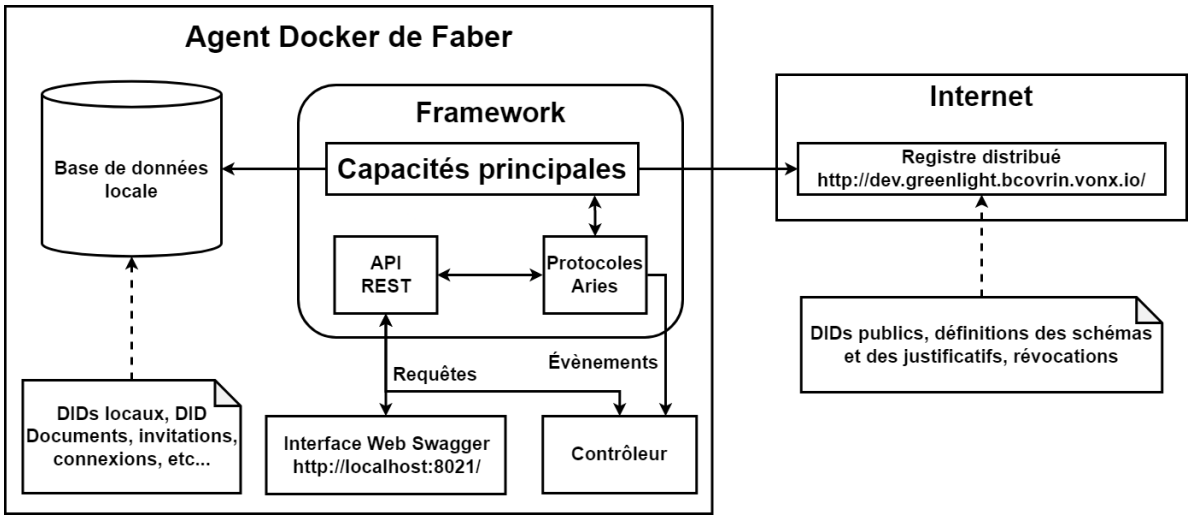


FIGURE 4.2 – *Architecture générée par l'agent de Faber*
Source: *de l'auteur*

Nous avons un conteneur *Docker* dans lequel s'exécute une instance du ***framework***. Nous communiquons avec celui-ci en envoyant des requêtes à l'***API*** au travers d'une interface ***Swagger*** accessible depuis le port prédéfini de l'agent sur l'adresse *IP* locale. Cette ***API*** utilise les différents protocoles pour transmettre des directives au ***framework***. Ceux-ci vont également communiquer des événements au contrôleur. Finalement, nous avons la base de données locale [6] ainsi que le registre distribué pour le stockage.

4.1.3 Scénario

Dans cette démonstration, nous incarnons un individu nommé Faber. Notre but est de créer une invitation depuis l'interface ***Swagger*** de son agent. Nous utilisons ensuite cette invitation pour établir une connexion vers le même agent.

6. Dans les différents schémas que nous utilisons, la base de données locale comprend aussi bien le stockage sur *Askar* que sur la mémoire.

4.1.4 Démonstration

Accès à l'interface *Swagger*

Nous accédons à l'interface ***Swagger*** via le port 8021 [7] de l'adresse locale. Le schéma 4.3 représente l'accès à l'interface grâce à un navigateur internet.

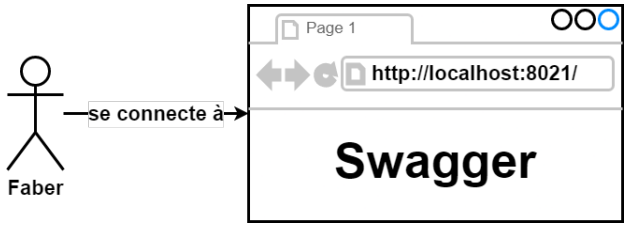


FIGURE 4.3 – *Accès à Swagger depuis un navigateur internet*
Source: *de l'auteur*

Une fois sur celle-ci, nous obtenons un aperçu de toutes les méthodes qu'expose l'instance du ***framework*** de l'agent de Faber. La figure 4.4 montre les fonctions disponibles pour la gestion des connexions.

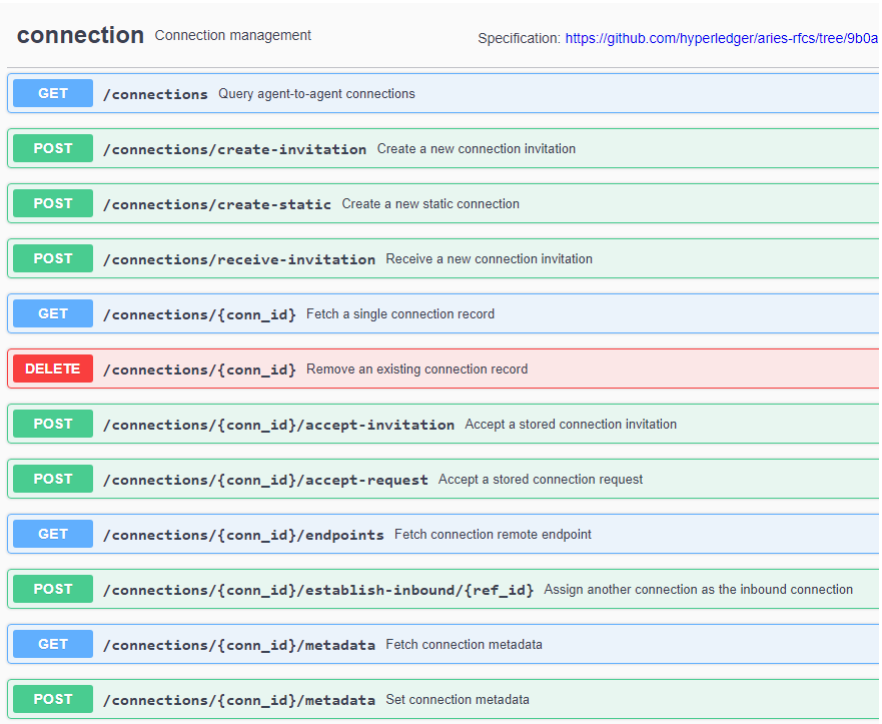


FIGURE 4.4 – *Échantillon de méthodes exposées sur Swagger*
Source: *de l'auteur*

Notons que chaque en-tête en dessus d'un groupe de fonctions, par exemple *"connection"*, représente le protocole qu'elles contactent lors de leur exécution.

7. Comme nous l'avons vu dans le prérequis, c'est le port défini pour l'agent de Faber.

Création d'une invitation

Le schéma 4.5 représente le processus lors de la création d'une invitation.

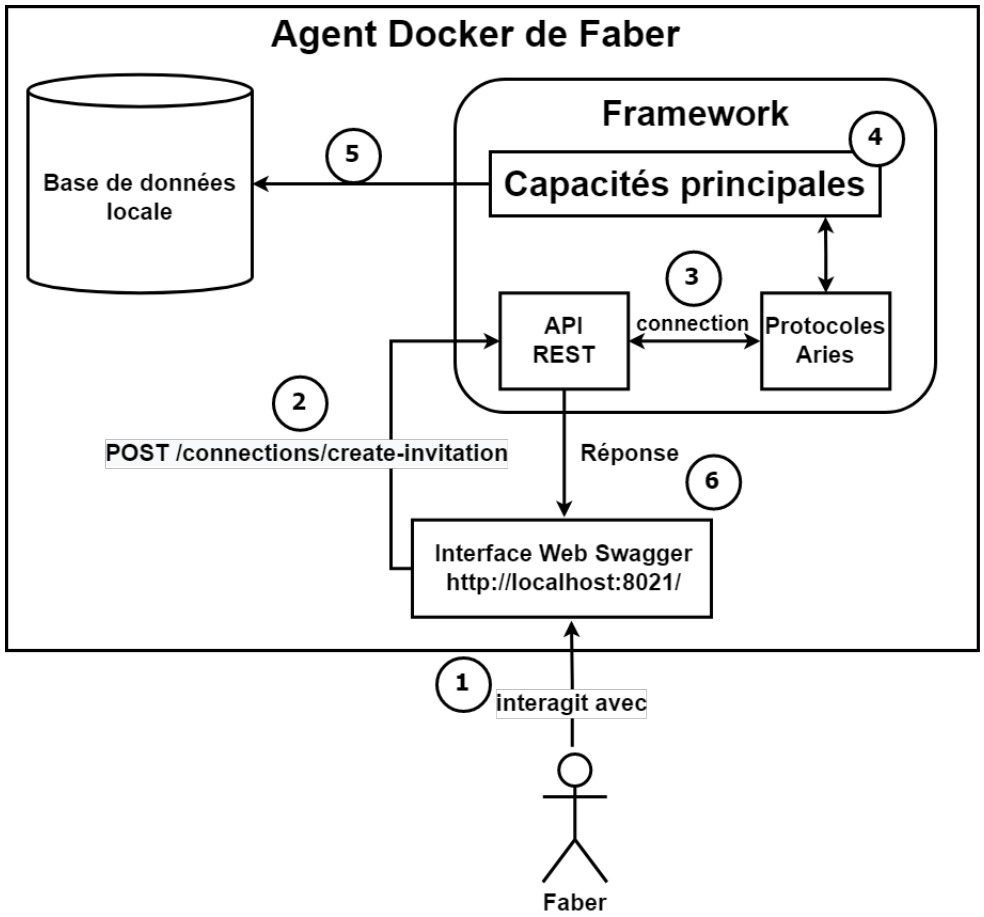


FIGURE 4.5 – *Processus de création d'une invitation avec Swagger*
Source: *de l'auteur*

Nous commençons par interagir **(1)** avec l'interface ***Swagger***. C'est la méthode *"POST /connections/create-invitation"* qui va permettre la création d'une invitation. L'exécution de cette méthode, pour cette démonstration, ne nécessite qu'un *"alias"*, nous utilisons la valeur *"POC1ALIAS"*. La figure 4.6 montre cette configuration.

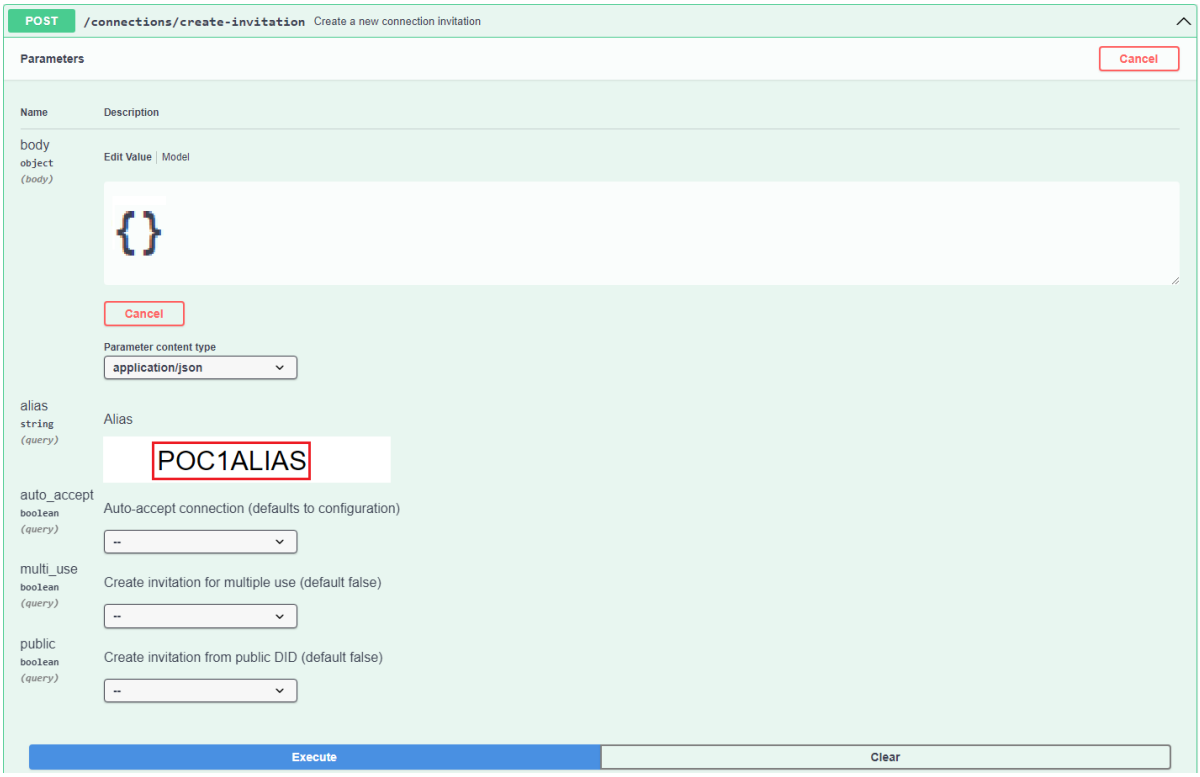


FIGURE 4.6 – *Création d'une invitation avec Swagger*
Source: *de l'auteur*

L'exécution envoie **(2)** une requête à l'***API*** qui contacte le protocole *"connection"* **(3)**. Le protocole transfère la demande au ***framework*** qui s'occupe de créer l'invitation **(4)** et de l'inscrire dans la base de données locale **(5)**. Les informations de l'invitation sont ensuite retransmises au protocole *"connection"* qui les transfère à l'***API*** pour que celle-ci nous renvoie une réponse sur l'interface ***Swagger*** **(6)** [8].


```
{
  "connection_id": "2bbed6d3-f12e-43ad-a1e6-94cee1cbc85d",
  "invitation": {
    "@type": "https://didcomm.org/connections/1.0/invitation",
    "@id": "cdacba3e-9e3c-4844-897c-934aca903901",
    "recipientKeys": [
      "CEDxGhcnpgxqnK3JE19kaMwLMJXgFzwY9jyQxha5EHg8"
    ],
    "label": "faber.agent",
    "serviceEndpoint": "http://192.168.65.3:8020"
  },
  "invitation_url": "http://192.168.65.3:8020?c_i=...",
  "alias": "POC1ALIAS"
}
```


Nous voyons le lien de l'invitation dans le champ *"invitation"* de la réponse obtenue.

8. Dans les prochains exemples, nous ne préciserons plus le chemin complet de retour de la réponse.

Réception d'une invitation et établissement d'une connexion

Le schéma 4.7 représente le processus lors de la réception d'une invitation et de l'établissement d'une connexion.

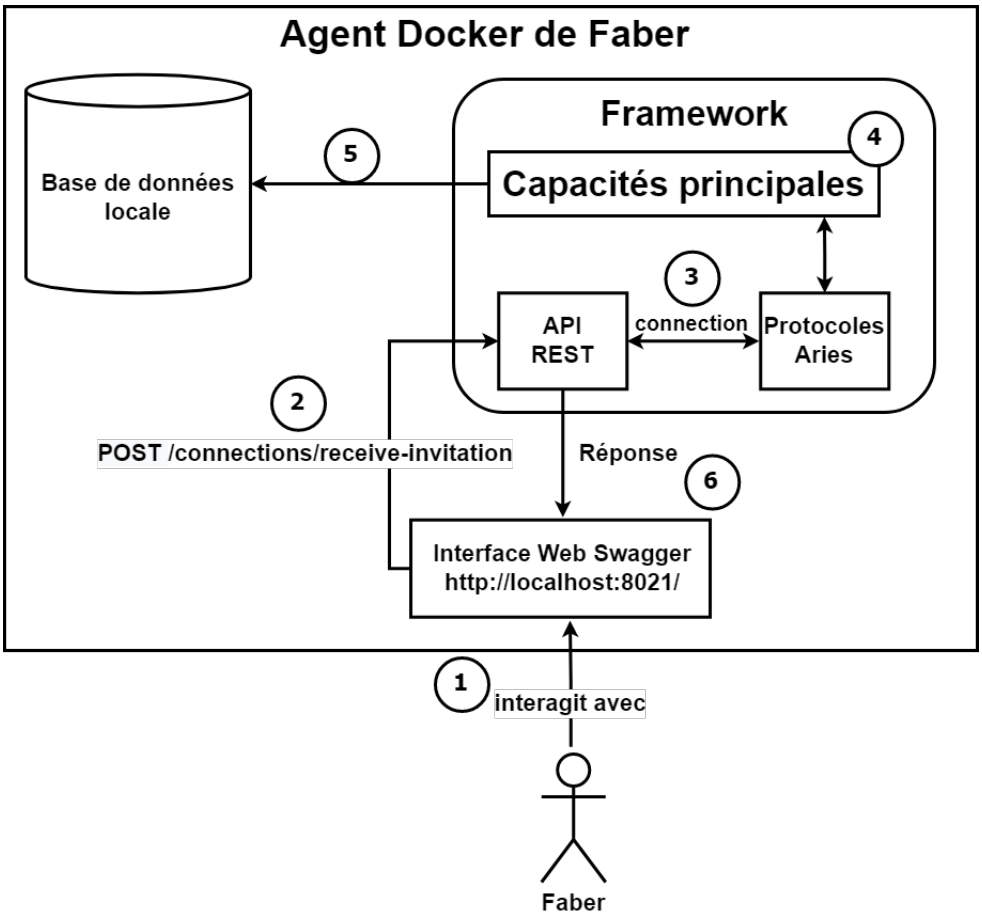


FIGURE 4.7 – *Processus de réception d'une invitation et de création d'une connexion avec Swagger*
Source: *de l'auteur*

Nous remarquons que le processus ressemble à celui de la création d'une invitation que nous avons vu dans la figure 4.6. Nous commençons par interagir **(1)** avec l'interface ***Swagger***. La méthode *"POST /connections/receive-invitation"* permet la réception d'une invitation. Pour l'exécuter, la requête doit contenir le lien [9] de l'invitation reçu précédemment.

```
{
    "@type": "https://didcomm.org/connections/1.0/invitation",
    "@id": "cdacba3e-9e3c-4844-897c-934aca903901",
    "recipientKeys": [
      "CEDxGhcnpgxqnK3JE19kaMwLMJXgFzwY9jyQxha5EHg8"
    ],
    "label": "faber.agent",
    "serviceEndpoint": "http://192.168.65.3:8020"
}
```

9. Pour rappel, c'est ce qui se trouve dans le champ *"invitation"* de la réponse reçue.

La figure 4.8 montre la configuration nécessaire avant l'exécution de la requête.

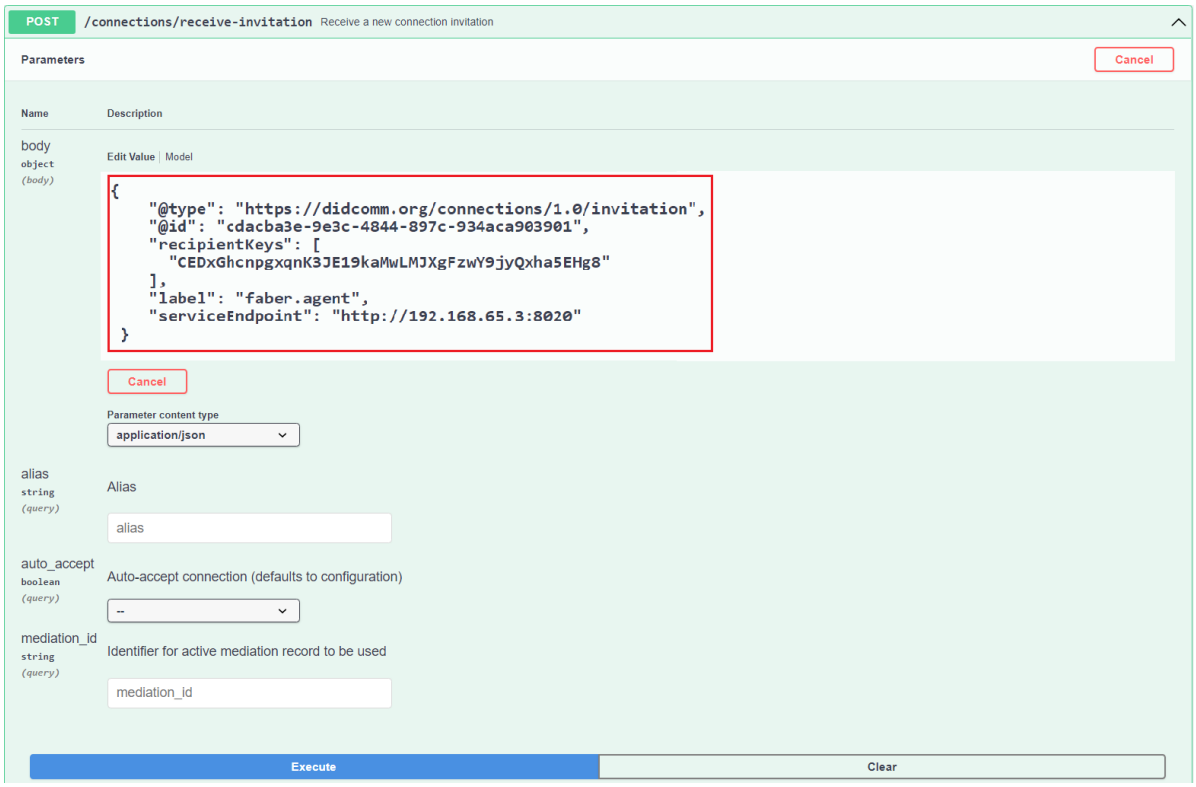


FIGURE 4.8 – *Réception d'une invitation avec Swagger*
Source: *de l'auteur*

L'exécution envoie **(2)** une requête à l'***API*** qui contacte le protocole **(3)**. Le protocole transfère la demande au ***framework*** qui gère **(4)** la réception de l'invitation, la création de la connexion et son inscription **(5)** dans la base de données locale. Les informations sur la connexion nous sont ensuite renvoyées sur l'interface ***Swagger*** **(6)**.


```
{
  "routing_state": "none",
  "invitation_msg_id": "cdacba3e-9e3c-4844-897c-934aca903901",
  "their_label": "faber.agent",
  "connection_protocol": "connections/1.0",
  "created_at": "2022-07-13T17:22:01.361328Z",
  "state": "request",
  "rfc23_state": "request-sent",
  "request_id": "92390169-0270-4d97-b00a-3e73dd3662eb",
  "accept": "auto",
  "my_did": "2k9ErKwWC8SWSxtan19DH4",
  "invitation_mode": "once",
  "updated_at": "2022-07-13T17:22:01.386303Z",
  "invitation_key": "CEDxGhcnpgxqnK3JE19kaMwLMJXgFzwY9jyQxha5EHg8",
  "connection_id": "1174a9d2-f471-490c-9857-67bf95d6972c",
  "their_role": "inviter"
}
```


La connexion est maintenant établie.

Récupération de la liste des connexions

Au fur et à mesure que nous avançons dans les *PoCs*, nous omettons volontairement certains éléments qui ne feraient que répéter ou complexifier certaines notions [10]. C'est le cas pour le schéma 4.9 qui représente le processus de récupération de la liste des connexions établies avec notre agent.

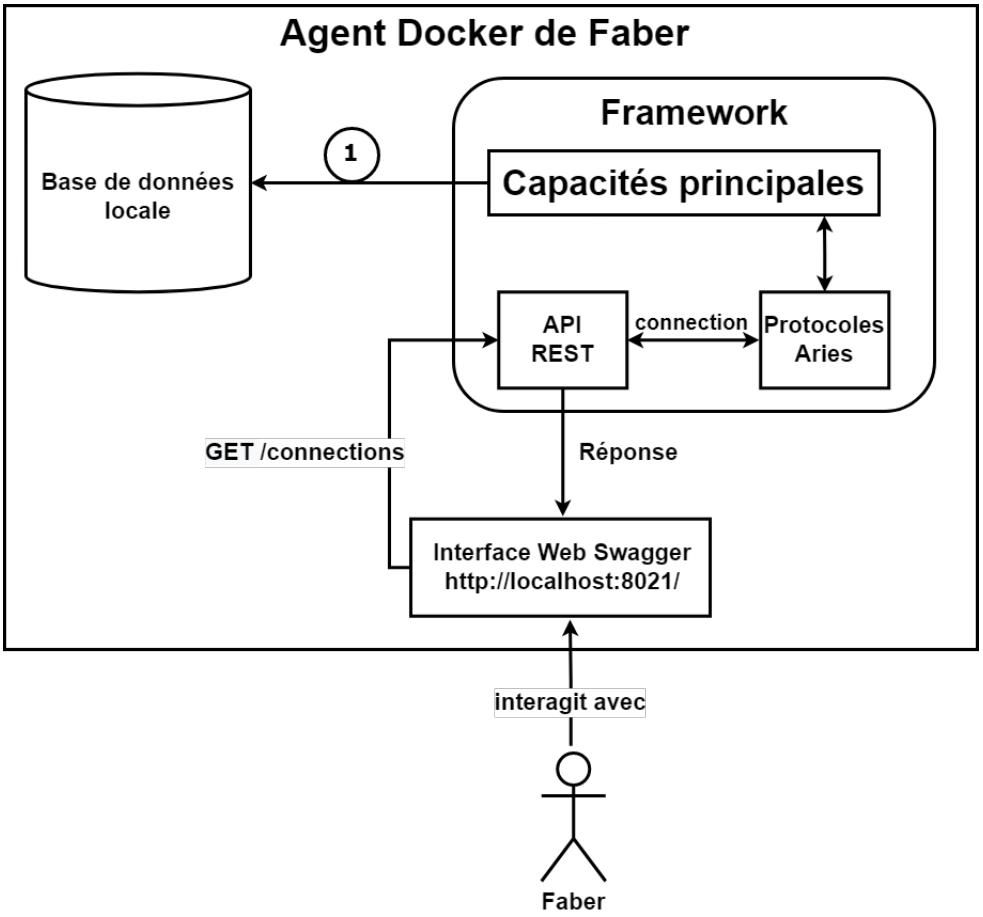


FIGURE 4.9 – *Processus de récupération des connexions avec Swagger*
Source: *de l'auteur*

La seule particularité de ce processus est que le ***framework*** va aller lire **(1)** ce qui se trouve dans le stockage local pour nous envoyer une réponse.

10. Pour rappel, le but de ces démonstrations est d'avoir un aperçu du fonctionnement du ***framework*** et non de comprendre toutes ces spécificités techniques.

```
{
  "results": [
    {
      "routing_state": "none",
      "alias": "POC1ALIAS",
      "their_label": "faber.agent",
      "connection_protocol": "connections/1.0",
      "created_at": "2022-07-13T15:23:17.441361Z",
      "state": "active",
      "rfc23_state": "completed",
      "accept": "auto",
      "my_did": "5miRwjNZotsNbUW3u1Rhhi",
      "invitation_mode": "once",
      "updated_at": "2022-07-13T17:22:01.501386Z",
      "invitation_key": "CEDxGhcnpgxqnK3JE19kaMwLMJXgFzwY9jyQxha5EHg8",
      "their_did": "2k9ErKwWC8SWSxtan19DH4",
      "connection_id": "2bbed6d3-f12e-43ad-a1e6-94cee1cbc85d",
      "their_role": "invitee"
    }
  ]
}
```


Nous obtenons ainsi une liste des différentes connexions établies avec l'agent de Faber. Le champ *"alias"* qui contient la valeur *"POC1ALIAS"* permet de confirmer la présence de la connexion que nous venons de créer. La valeur *"active"* du champ *"state"* indique également que la connexion est actuellement active.

Une connexion est un élément primordial lors de l'utilisation du ***framework***. C'est elle qui permet à des agents de communiquer entre eux en utilisant les fonctionnalités offertes, comme l'échange de *Verifiable Credentials*, par le *Cloud Agent Python*.

Nettoyage

Pour éviter des conflits lors du *PoC* suivant, il est important de supprimer les éléments créés sur *Docker* pendant cette démonstration. Pour ce faire, nous utilisons les commandes [11] suivantes.

```
1 valonrexhepi@WINDOWSDESKTOP:~$ docker stop $(docker ps -a -q)
2 valonrexhepi@WINDOWSDESKTOP:~$ docker rm $(docker ps -a -q)
3 valonrexhepi@WINDOWSDESKTOP:~$ docker rmi $(docker images -a -q)
4 valonrexhepi@WINDOWSDESKTOP:~$ docker system prune -a -f
```

11. Attention, ces commandes suppriment tous les conteneurs, images et volumes de *Docker*. Plus d'informations sont disponibles sur la documentation officielle.

4.1.5 Résultats

Lors de ce *PoC*, nous avons commencé par lancer un agent prédéfini. Ce lancement nous a permis d'établir certaines bases de compréhension et de connaissances qui seront utiles pour la démonstration suivante.

Une fois l'agent lancé, nous avons accédé à l'interface ***Swagger*** qui nous a permis d'avoir un aperçu sur les différentes fonctions disponibles sur le ***framework***.

Avec l'aide de cette interface, nous avons créé une invitation qui nous a ensuite permis d'établir une connexion avec un agent. Dans cette démonstration, le même agent est utilisé pour toutes ces opérations, et la connexion est établie vers lui-même. Cependant, le processus est semblable lorsqu'il y a deux agents, comme nous allons le voir dans le prochain *PoC*.

La réalisation de cette démonstration ne nécessite pas de connaissances particulières sur le fonctionnement du ***framework***. Elle permet une introduction rapide à l'*Aries Cloud Agent Python* et à ses fonctionnalités. Finalement, la documentation pour sa mise en place est bien maintenue par la communauté *Hyperledger* et permet de ce fait une compréhension et une reproduction sans problème.

4.2 Développement d'un contrôleur en Python et échanges de justificatifs

Dans cette démonstration, nous reproduisons le *lab* suivant sur une machine ***Wsl Ubuntu*** s'exécutant sur ***Windows*** 11. Pour cette exemple, nous utilisons l'*Aries Cloud Agent Python* ainsi que *Docker* et *Git*.

4.2.1 Prérequis

Nous continuons d'utiliser l'archive du *Cloud Agent Python* clonée dans le *PoC* précédent. Dans cette démonstration, nous travaillons avec trois agents différents. Il y a celui de Faber, celui d'Alice et celui d'une entreprise nommée ACME. Pour rappel, ce sont des agents préconfigurés mis à notre disposition pour tester l'utilisation du ***framework***.

Avant de continuer, nous revenons sur un code rencontré dans le premier *PoC*.

```
python -m demo.runners.$AGENT_NAME $@
```

Pour rappel, cette ligne exécute le contrôleur [12] de l'agent spécifié lors du lancement du conteneur. Le dossier *"demo/runners/"* contient les différents contrôleurs de démonstration disponibles du ***framework***.

```
valonrexhepi@WINDOWSDESKTOP:~/aries-cloudagent-python/demo/runners$ ls
acme.py  agent_container.py  alice.py  faber.py
```

Nous voyons ici quatre fichiers différents, chacun représentant le contrôleur de l'agent dont il porte le nom. L'extension *".py"* indique que langage de programmation utilisé est le Python [13] Il y a cependant une spécificité. L'*"agent_container.py"* est la classe principale dont héritent [14] les trois autres. La figure 4.10 schématise ce concept.

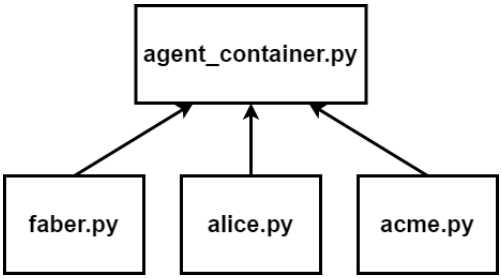


FIGURE 4.10 – *Héritage des contrôleurs sur l'Aries Cloudagent Python*
Source: *de l'auteur*

12. Un contrôleur, comme nous l'avons vu dans le chapitre 2.3.1, est une ***classe*** qui permet de définir la logique métier de l'agent.
13. Pour rappel, n'importe quel langage peut être utilisé tant que celui-ci permet de faire des requêtes ***REST***.
14. La notion d'héritage est un principe de programmation qui permet de créer des classes à partir d'autres classes.

La classe principale sert uniquement à définir des éléments, comme des méthodes, qui seront utilisés par les contrôleurs qui héritent de celle-ci. Le schéma 4.11 montre comment le fichier *"alice.py"* est construit.

FIGURE 4.11 – *Exemple du contenu d'un contrôleur*
Source: *de l'auteur*

Chaque contrôleur contient une partie *"imports"* qui déclare les librairies utilisées, une définition de classe dans laquelle différents attributs sont définis et une méthode *"main"* qui est le point d'entrée de l'exécution du contrôleur. Dans cette figure, nous voyons également une méthode nommée *"input_invitation"* qui définit le fonctionnement de l'agent d'Alice lorsqu'une invitation est reçue. Nous n'entrons pas plus dans les détails de ces contrôleurs car les éléments expliqués suffisent à la compréhension de la suite de ce *PoC*.

Contrairement à la démonstration précédente, nous ne lançons pas directement les différents agents avec la commande *"./run_demo"*. Le but est d'abord de développer les parties manquantes du contrôleur de l'entreprise ACME. Une fois ce développement terminé, nous démarrons trois agents : Faber, Alice et l'ACME. Pour interagir avec les contrôleurs, nous utilisons l'interface en ligne de commandes, aussi appelée *"terminal"*, générée lors du lancement des conteneurs. Notons qu'elle est différente en fonction des agents.

L'aperçu de l'interface de l'ACME permet une meilleure compréhension de ce que nous réalisons par la suite.

```
# INTERFACE DE L'ACME
    (1) Issue Credential
    (2) Send Proof Request
    (3) Send Message
    (X) Exit?
    [1/2/3/X]
```

Nous voyons qu'elle se compose d'options permettant d'effectuer des actions. Par exemple, l'option **[2]** va indiquer au contrôleur que l'ACME souhaite envoyer une demande de ***proof***. Ces options correspondent à des opérations définies dans la fonction *"main"* du contrôleur de l'ACME comme représenté par la figure 4.12.

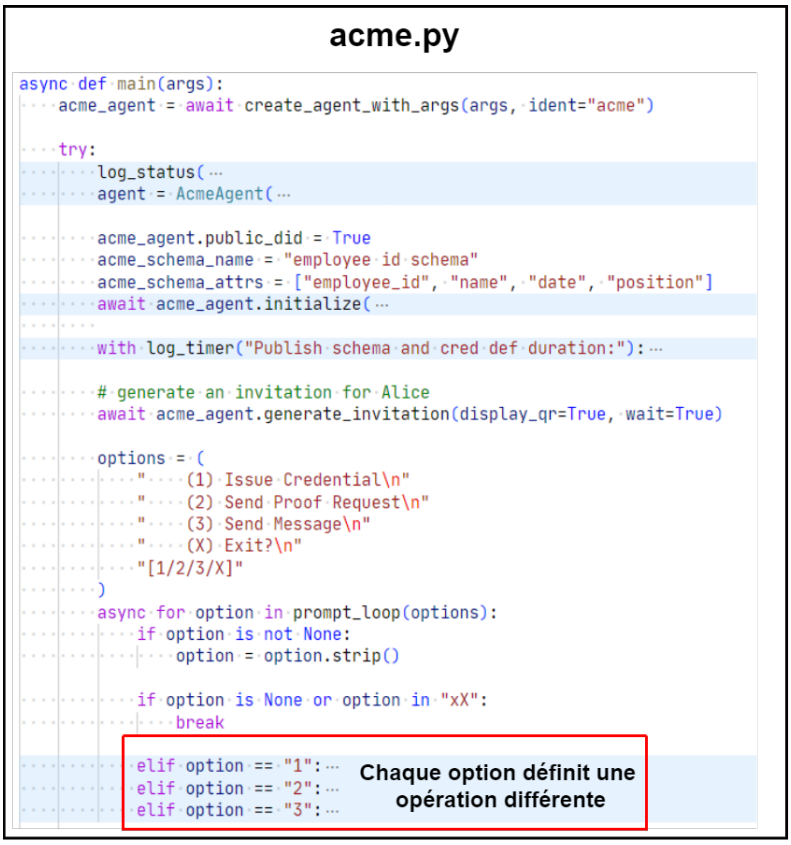


FIGURE 4.12 – *Choix des options du contrôleur de l'ACME*
Source: *de l'auteur*

Nous développons certaines de ces opérations dans ce *PoC*.

4.2.2 Scénario

Le scénario consiste à la mise en place du triangle de confiance représenté par la figure 4.13.

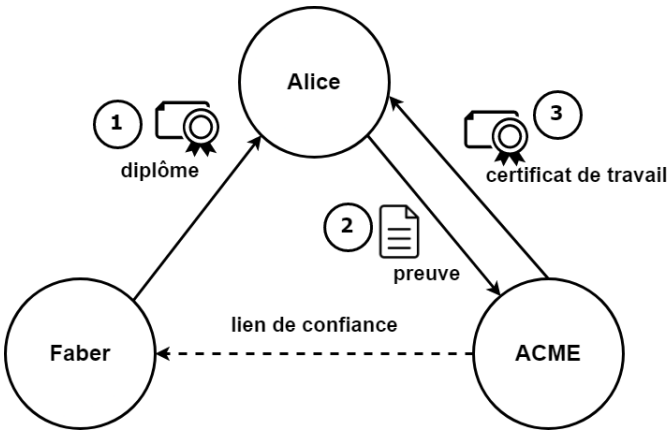


FIGURE 4.13 – *Triangle de confiance entre Faber, Alice et l'ACME*
Source: *de l'auteur*

Nous voulons qu'Alice obtienne un certificat de travail de l'entreprise ACME. Pour cela, Faber envoie **(1)** un justificatif représentant un diplôme à Alice. Alice utilise ce justificatif pour prouver **(2)** à l'entreprise ACME qu'elle est diplômée. Enfin, après validation, l'entreprise lui transmet **(3)** un justificatif de travail.

4.2.3 Démonstration du développement du contrôleur de l'ACME

Développement de la demande de preuve

Nous développons d'abord l'option **[2]** du contrôleur. Pour ce faire, nous ouvrons le fichier *"acme.py"* et commençons par lui ajouter [15] les imports suivants.

```
import random # permet de générer des nombres aléatoires

from datetime import date # permet l'utilisation des dates
from uuid import uuid4 # permet de générer des identifiants uniques
```

Ensuite, nous modifions la valeur de la variable *"CRED_PEVIEW_TYPE"* par le code suivant.

```
CRED_PREVIEW_TYPE = (
    "did:sov:BzCbsNYhMrjHiqZDTUASHg;spec/issue-credential/2.0/credential-preview"
) # permet de définir le type de justificatif utilisé
```

Comme nous pouvons le voir, le *DID* est récupéré depuis la solution *Sovrin*.

15. Pour rappel, la figure 4.11 permet de voir les différentes parties d'un contrôleur.

Nous développons ensuite le code lancé lorsque l'option [16] **[2]**, pour la demande d'une preuve, est choisie. C'est la fonction utilisée pour la relation **(2)** du schéma 4.13.

```
elif option == "2":
    log_status("#20 Request proof of degree from alice")
    # (1) req_attrs définit les attributs que nous demandons dans la proof
    req_attrs = [
        {
            "name": "name",
            "restrictions": [{"schema_name": "degree schema"}]
        },
        {
            "name": "date",
            "restrictions": [{"schema_name": "degree schema"}]
        },
        {
            "name": "degree",
            "restrictions": [{"schema_name": "degree schema"}]
        }
    ]
    # (2) création de la demande
    indy_proof_request = {
        "name": "Proof of Education",
        "version": "1.0",
        "nonce": str(uuid4().int),
        "requested_attributes": {
            f"0_{req_attr['name']}_uuid": req_attr for req_attr in req_attrs
        },
        "requested_predicates": {}
    }
    proof_request_web_request = {
        "connection_id": agent.connection_id,
        "presentation_request": {"indy": indy_proof_request},
    }
    # (3) envoie de la demande au travers d'une requête REST
    await agent.admin_POST(
        "/present-proof-2.0/send-request",
        proof_request_web_request
    )
```

Dans ce code, nous distinguons trois étapes. La première étape **(1)** définit, dans une variable *"req_attrs"*, les attributs que doit contenir la ***proof*** demandée, c'est-à-dire un nom, une date et un diplôme. Ces attributs font également échos au schéma inscrit, voir la figure 4.1, lors du lancement de l'agent de Faber dans le *PoC* précédent [17]. Dans la seconde étape **(2)**, le contenu de la demande de preuve est construit. Finalement, dans la dernière étape **(3)**, la requête est envoyée en contactant l'***API*** au travers d'une requête ***REST***[18].

16. Pour rappel, l'emplacement de ces options peut être visualisé sur le schéma 4.12.
17. Cela confirme que le justificatif qu'envoie Faber à Alice contient les éléments recherchés par l'ACME.
18. Dans le *PoC* précédent, la requête se faisait manuellement en indiquant des valeurs de paramètres dans les méthodes de l'interface ***Swagger***. La différence désormais est que c'est le contrôleur qui exécute et définit le contenu de la requête.

Développement de la réception de la preuve

Nous développons la méthode *"handle_present_proof_v2_0"* du contrôleur. Celle-ci est appelée lorsque l'ACME reçoit une réponse à sa demande de ***proof***.

```
async def handle_present_proof_v2_0(self, message):
    state = message["state"]
    pres_ex_id = message["pres_ex_id"]
    self.log(f"Presentation: state = {state}, pres_ex_id = {pres_ex_id}")

    if state == "presentation-received":
        log_status("#27 Process the proof provided by X")
        log_status("#28 Check if proof is valid")

        # (1) une preuve a été reçue et est vérifiée
        proof = await self.admin_POST(
            f"/present-proof-2.0/records/{pres_ex_id}/verify-presentation"
        )
        self.log("Proof = ", proof["verified"])
        pres_req = message["by_format"]["pres_request"]["indy"]
        pres = message["by_format"]["pres"]["indy"]
        is_proof_of_education = pres_req["name"] == "Proof of Education"

        # (2) si cette preuve est un diplôme alors les attributs sont vérifiés et affichés
        if is_proof_of_education:
            log_status("#28.1 Received proof of education, check claims")
            for (referent, attr_spec) in pres_req["requested_attributes"].items():
                self.log(
                    f"{attr_spec['name']}: "
                    f"{pres['requested_proof']['revealed_attrs'][referent]['raw']}"
                )
            for id_spec in pres["identifiers"]:
                self.log(f"schema_id: {id_spec['schema_id']}")
                self.log(f"cred_def_id {id_spec['cred_def_id']}")
        else:
            self.log("#28.1 Received ", message["presentation_request"]["name"])
```

Dans ce code, nous constatons que lorsqu'une preuve est reçue sa validité est confirmée avec un appel **(1)** à l'***API***. Ensuite, si celle-ci représente bien un diplôme, les attributs sont vérifiés et affichés **(2)** sur le terminal.

Développement de l'échange de justificatif

Pour que l'ACME envoie un justificatif de travail, il faut définir sa structure. Pour ce faire, nous définissons un nouveau schéma [19] dans le contrôleur.

```
acme_schema_name = "employee id schema" # nom du schéma
acme_schema_attrs = ["employee_id", "name", "date", "position"] # attributs du schéma

# initialisation des variables du contrôleur
await acme_agent.initialize(
    the_agent=agent, # nom de l'agent --> "acme"
    schema_name=acme_schema_name, # nom de son schéma --> "employee id schema"
    schema_attrs=acme_schema_attrs, # attributs --> ["employee_id", "name", "date",
    ↪  "position"]
)
```

Nous voyons que le justificatif est constitué d'un nom, d'une date et de la position occupée dans l'entreprise. Comme lorsque nous lançons l'agent de Faber dans le *PoC* précédent, le contrôleur doit demander l'écriture de la définition du schéma et du justificatif dans le registre distribué. Nous ajoutons le code de cette opération sous l'initialisation des variables du contrôleur.

```
# initialisation des variables du contrôleur
await acme_agent.initialize(
    # ...
)
# ...
with log_timer("Publish schema and cred def duration:"):
    # (1) une version du schéma est générée
    version = format(
        "%d.%d.%d"
        % (
            random.randint(1, 101),
            random.randint(1, 101),
            random.randint(1, 101),
        )
    )
    # (2) une requête est envoyée permettant l'inscription de la définition du schéma et du
    ↪  justificatif sur le registre distribué
    (schema_id, cred_def_id) = await agent.register_schema_and_creddef(
        "employee id schema",
        version,
        ["employee_id", "name", "date", "position"],
        support_revocation=False,
        revocation_registry_size=TAILS_FILE_COUNT,
    )
```

Nous voyons qu'une version du schéma est aléatoirement générée **(1)** et que le contrôleur demande l'inscription **(2)** de sa définition sur le registre distribué.

19. Le code est déjà présent dans le contrôleur mais il est en commentaire. Il faut ainsi décommenter les lignes concernées.

Ensuite, nous développons l'option **[1]** du terminal. Elle actionne l'envoi d'une offre de justificatif de travail à Alice.

```
elif option == "1":
    log_status("#13 Issue credential offer to X")

    # (1) les valeurs des attributs du justificatif sont définies
    agent.cred_attrs[cred_def_id] = {
        "employee_id": "ACME0009",
        "name": "Alice Smith",
        "date": date.isoformat(date.today()),
        "position": "CEO",
    }

    # (2) création du contenu du justificatif
    cred_preview = {
        "@type": CRED_PREVIEW_TYPE,
        "attributes": [
            {"name": n, "value": v} for (n, v) in agent.cred_attrs[cred_def_id].items()
        ],
    }

    # (3) création de l'offre avec les détails du justificatif
    offer_request = {
        "connection_id": agent.connection_id,
        "comment": f"Offer on cred def id {cred_def_id}",
        "credential_preview": cred_preview,
        "filter": {"indy": {"cred_def_id": cred_def_id}},
    }

    # (4) envoi de l'offre
    await agent.admin_POST("/issue-credential-2.0/send-offer", offer_request)
```

Ce code définit **(1)** les valeurs des attributs du justificatif qui est envoyé à Alice. Nous voyons qu'il est générée **(2)** avec ces attributs. L'offre est ensuite créée **(3)** et envoyée **(4)** à l'***API***.

La dernière partie que nous développons est celle qui permet l'envoi du justificatif à Alice lorsqu'elle accepte l'offre reçue. Celle-ci est ajoutée dans la méthode *"handle_issue_credential_v2_0"*.

```
async def handle_issue_credential_v2_0(self, message):
    # ...
    if state == "request-received":
        if not message.get("auto_issue"):
            # envoi du justificatif de travail
            await self.admin_POST(
                f"/issue-credential-2.0/records/{cred_ex_id}/issue",
                {"comment": f"Issuing credential, exchange {cred_ex_id}"},
            )
```

Le développement effectué durant cette section permet de créer la relation **(3)** du schéma 4.13. Nous pouvons constater que cet échange s'effectue en deux étapes : l'envoi d'une offre de justificatif et l'envoi du justificatif.

4.2.4 Démonstration du scénario

Avant de commencer, notons que pendant cette démonstration nous ne précisons pas tous les détails de l'exécution des différentes opérations. Le but est de comprendre le fonctionnement général du ***framework*** *Cloud Agent Python*. De plus, les différentes notions rencontrées durant les sections précédentes permettent de comprendre le fonctionnement de certains aspects sans toutefois les détailler.

Connexion entre Faber et Alice

Dans cette section, nous établissons une connexion entre l'agent de Faber et d'Alice. Nous commençons par lancer l'agent de Faber.

```
# Terminal de Faber
valonrexhepi@WINDOWSDESKTOP:~/aries-cloudagent-python/demo$
↪  LEDGER_URL=http://dev.greenlight.bcovrin.vonx.io ./run_demo faber
...
Faber      DID: AqRhLSCqnPcCpsW9W1bouU # DID public de Faber
...
Schema ID: AqRhLSCqnPcCpsW9W1bouU:2:degree schema:57.2.98
Cred def ID: AqRhLSCqnPcCpsW9W1bouU:3:CL:142837:faber.agent.degree_schema
Publish schema/cred def duration: 10.15s # publication des définitions sur le registre
↪  distribué
```

Ce lancement génère le processus représentée par la figure 4.14.

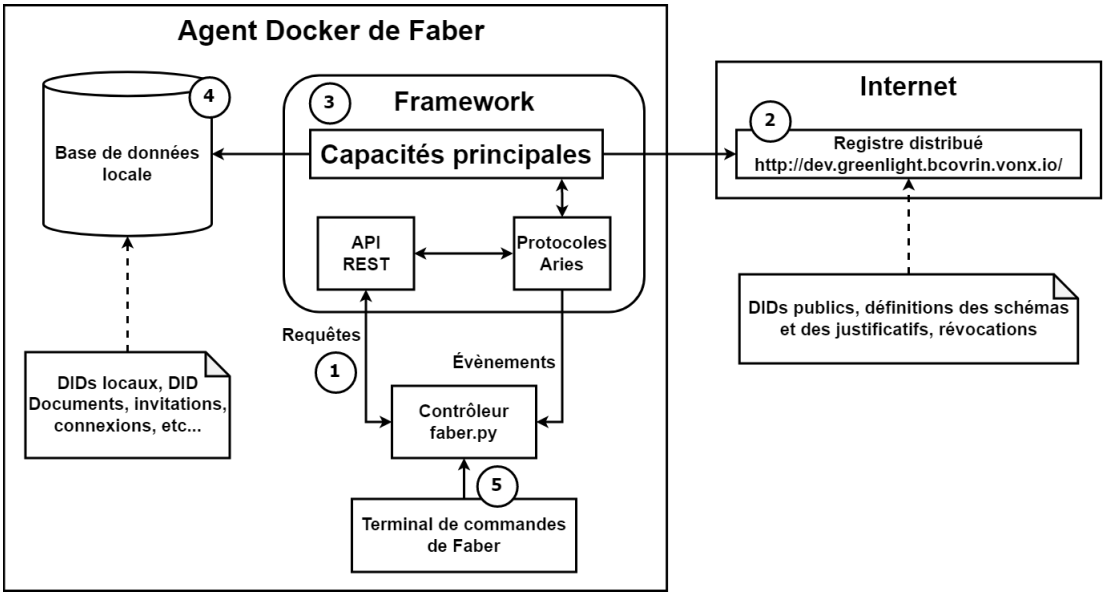


FIGURE 4.14 – *Processus de lancement de l'agent de Faber*
Source: *de l'auteur*

Le contrôleur *"faber.py"* envoie **(1)** des requêtes [20] à l'***API***. Ces requêtes permettent l'inscription **(2)** du *DID public "AqRhLSCqnPcCpsW9W1bouU"*, ainsi que la définition du schéma, de Faber sur le registre distribué [21]. Une invitation est générée **(3)** automatiquement et est stockée dans la base de données locale. Le lien de cette invitation nous est ensuite transmis **(5)** sur le terminal de Faber.

```
# Terminal de Faber
# Lien de l'invitation
Invitation Data:
{"@type": "https://didcomm.org/out-of-band/1.0/invitation", "@id":
↪  "2c3b2185-6c58-491b-9a0e-a0e0bf20bb8f", "label": "faber.agent", "services": [{"id":
↪  "#inline", "type": "did-communication", "recipientKeys":
↪  ["did:key:z6MkfoYzEifMrknrSk3ZbEZoMv7hTdfjVKgEvoLQoMT9oSoS"], "serviceEndpoint":
↪  "http://192.168.65.3:8020"}], "handshake_protocols":
↪  ["https://didcomm.org/didexchange/1.0"]}

# Nous fournissions une version formatée de ce lien pour une meilleure lecture
{
   "@type":"https://didcomm.org/out-of-band/1.0/invitation",
   "@id":"2c3b2185-6c58-491b-9a0e-a0e0bf20bb8f",
   "label":"faber.agent",
   "services":[
      {
         "id":"#inline",
         "type":"did-communication",
         "recipientKeys":[
            "did:key:z6MkfoYzEifMrknrSk3ZbEZoMv7hTdfjVKgEvoLQoMT9oSoS" # DID privée
         ],
         "serviceEndpoint":"http://192.168.65.3:8020"
      }
   ],
   "handshake_protocols":[
      "https://didcomm.org/didexchange/1.0"
   ]
}
```

Un autre agent peut utiliser ce lien pour établir une connexion, comme dans la section 4.1.4 du *PoC* précédent.

Le contrôleur d'Alice, contrairement à celui de Faber, ne crée pas d'invitation ou de schémas. Cependant, il attend la réception d'une invitation de Faber, comme nous l'indique le terminal.

```
# Terminal d'Alice
valonrexhepi@WINDOWSDESKTOP:~/aries-cloudagent-python/demo$
↪  LEDGER_URL=http://dev.greenlight.bcovrin.vonx.io ./run_demo alice
... # il n'y a ni DID public, ni de schémas
# En attente de l'invitation de faber.py
Invite details:
```

20. Nous ne citons pas toutes les requêtes appelées car elles sont disponibles dans le code du contrôleur.
21. Nous pouvons vérifier l'inscription de ces éléments depuis l'historique des transactions du registre distribué. Ce lien renvoie vers les transactions concernées.

Le schéma 4.15 représente le processus de connexion d'Alice à Faber.

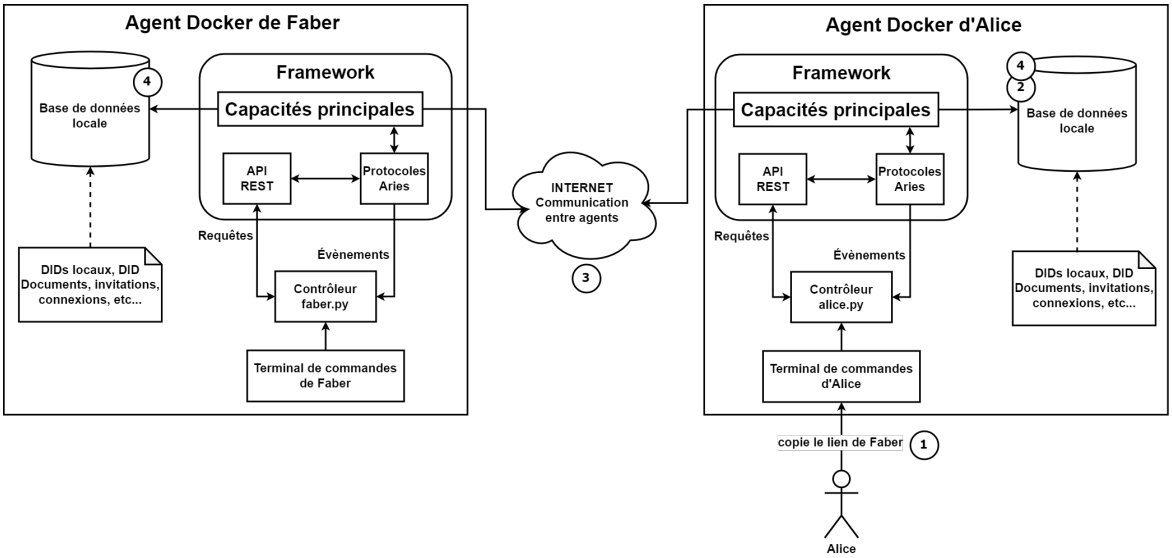


FIGURE 4.15 – *Processus de connexion entre Faber et Alice*
Source: *de l'auteur*

Alice interagit **(1)** avec son terminal en copiant le lien d'invitation reçu de Faber.

```
# Terminal d'Alice
# En attente de l'invitation de faber.py
Invite details: {"@type": "https://didcomm.org/out-of-band/1.0/invitation", "@id":
↪  "2c3b2185-6c58-491b-9a0e-a0e0bf20bb8f", "label": "faber.agent", "services": [{"id":
↪  "#inline", "type": "did-communication", "recipientKeys":
↪  ["did:key:z6MkfoYzEifMrknrSk3ZbEZoMv7hTdfjVKgEvoLQoMT9oSoS"], "serviceEndpoint":
↪  "http://192.168.65.3:8020"}], "handshake_protocols":
↪  ["https://didcomm.org/didexchange/1.0"]}
```

L'invitation reçue est inscrite **(2)** sur la base de données locale. L'agent de Faber est averti **(3)** de cette réception. La connexion est ensuite établie et les deux agents la stockent **(4)** localement.

La figure 4.16 représente l'architecture lorsque les deux agents sont connectés.

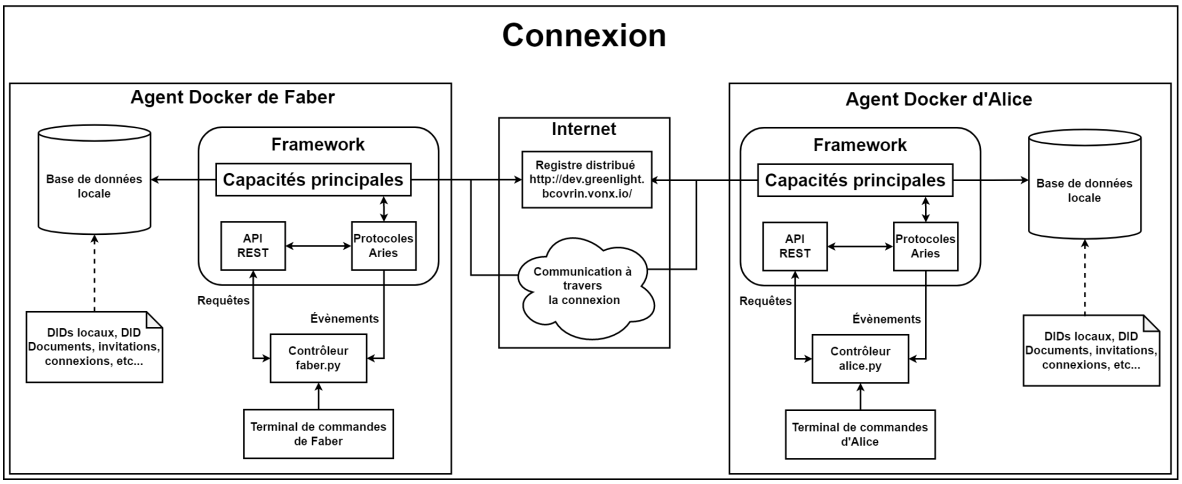


FIGURE 4.16 – *Architecture après la connexion de l'agent de Faber et d'Alice*
Source: *de l'auteur*

Les terminaux de Faber et Alice affichent maintenant les différentes options de communication disponibles.

```
# Terminal de Faber
Faber      | Connected
Faber      | Check for endorser role ...
    (1) Issue Credential
    (2) Send Proof Request
    (2a) Send *Connectionless* Proof Request (requires a Mobile client)
    (3) Send Message
    (4) Create New Invitation
    (T) Toggle tracing on credential/proof exchange
    (X) Exit?
[1/2/3/4/T/X]
# -------------------------------------------------------------------------
# Terminal d'Alice
Alice      | Connected
Alice      | Check for endorser role ...
Connect duration: 0.19s
    (3) Send Message
    (4) Input New Invitation
    (X) Exit?
[3/4/X]
```

Échange d'un justificatif entre Faber et Alice

Cette étape correspond à la relation **(1)** du triangle de confiance entre Faber, Alice et l'ACME. Pour rappel, le schéma complet est représenté par la figure 4.13.

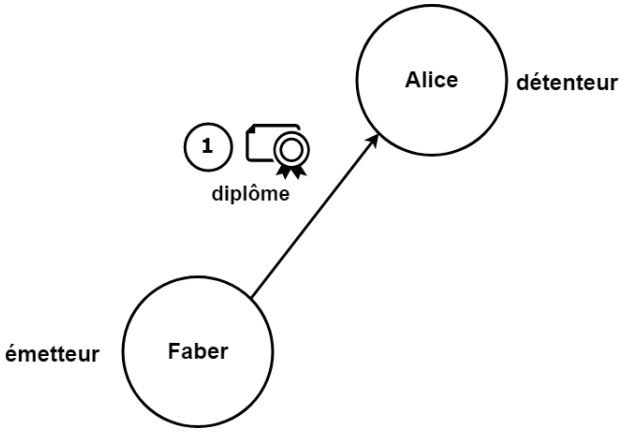


FIGURE 4.17 – *Première relation du triangle de confiance*
Source: *de l'auteur*

La figure 4.18 représente le processus d'envoi du justificatif.

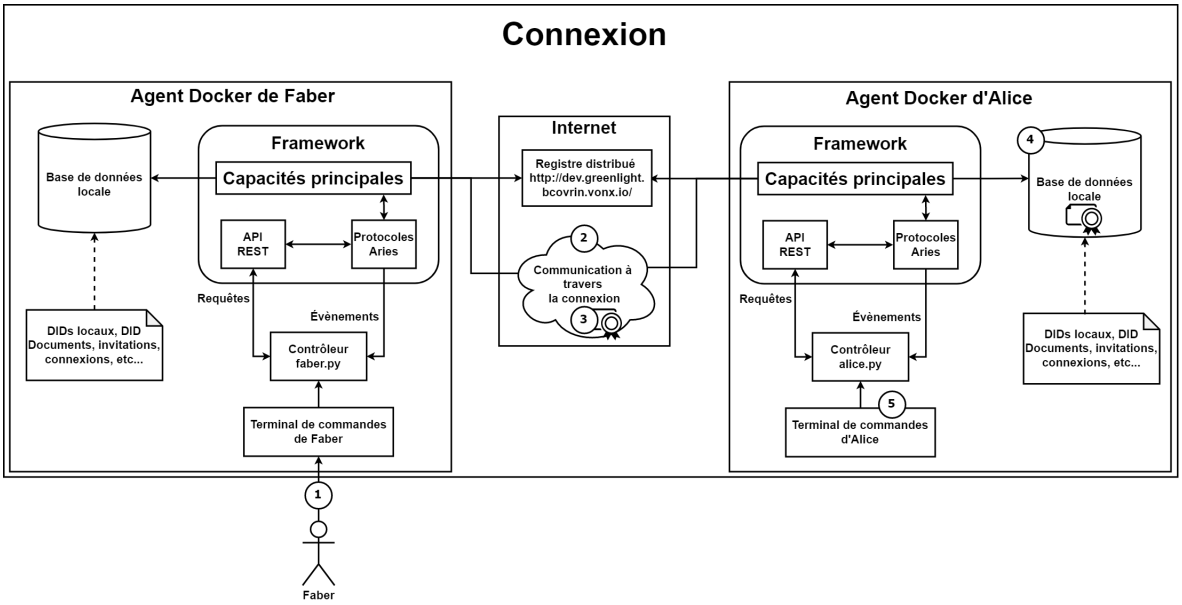


FIGURE 4.18 – *Processus d'échange d'un justificatif entre Faber et Alice*
Source: *de l'auteur*

Pour l'échange, Faber commence par choisir **(1)** l'option **[1]** sur son terminal, puis pour la suite de ce *PoC* nous pouvons stopper son agent avec l'option **[x]**.

```
# Terminal de Faber
[1/2/3/4/T/X] 1
# Envoi de l'offre de justificatif à Alice
Faber      | Credential: state = offer-sent, cred_ex_id =
↪  95d18f94-6d25-48fe-99f8-6e54609594ea

# Offre acceptée par Alice
Faber      | Credential: state = request-received, cred_ex_id =
↪  95d18f94-6d25-48fe-99f8-6e54609594ea

# Envoi du justificatif à Alice
Faber      | Credential: state = credential-issued, cred_ex_id =
↪  95d18f94-6d25-48fe-99f8-6e54609594ea
Faber      | Credential: state = done, cred_ex_id = 95d18f94-6d25-48fe-99f8-6e54609594ea
    (1) Issue Credential
    (2) Send Proof Request
    (2a) Send *Connectionless* Proof Request (requires a Mobile client)
    (3) Send Message
    (4) Create New Invitation
    (T) Toggle tracing on credential/proof exchange
    (X) Exit?
[1/2/3/4/T/X] x
Shutting down agent ...
Faber      |
Faber      | Shutting down
Faber      | Exited with return code 0
```

Une offre de justificatif est envoyée **(2)** à Alice à travers la connexion établie. Si Alice accepte, le diplôme lui est transmis **(3)**. L'agent d'Alice stocke **(4)** ensuite le justificatif localement. Les différentes opérations s'affichent **(5)** sur le terminal d'Alice.

```
# Terminal d'Alice
# Réception de l'offre
Alice      | Credential: state = offer-received, cred_ex_id =
↪  1520055c-2f9a-4aa1-a57b-23326e763e5c
#15 After receiving credential offer, send credential request
Alice      | Credential: state = request-sent, cred_ex_id =
↪  1520055c-2f9a-4aa1-a57b-23326e763e5c

# Réception du diplôme
Alice      | Credential: state = credential-received, cred_ex_id =
↪  1520055c-2f9a-4aa1-a57b-23326e763e5c

# Stockage du justificatif
Alice      | Credential: state = done, cred_ex_id = 1520055c-2f9a-4aa1-a57b-23326e763e5c
Credential details:
  {
    "referent": "7996c30e-6ff5-4206-b731-8786c2c743d1",
    "schema_id": "AqRhLSCqnPcCpsW9W1bouU:2:degree schema:57.2.98",
    "cred_def_id": "AqRhLSCqnPcCpsW9W1bouU:3:CL:142837:faber.agent.degree_schema",
    "rev_reg_id": null,
    "cred_rev_id": null,
    "attrs": {
```

```
      "name": "Alice Smith",
      "date": "2018-05-28",
      "timestamp": "1657784044",
      "degree": "Maths",
      "birthdate_dateint": "19980714"
    }
  }

Alice      | credential_id 7996c30e-6ff5-4206-b731-8786c2c743d1
Alice      | cred_def_id AqRhLSCqnPcCpsW9W1bouU:3:CL:142837:faber.agent.degree_schema
Alice      | schema_id AqRhLSCqnPcCpsW9W1bouU:2:degree schema:57.2.98
    (3) Send Message
    (4) Input New Invitation
    (X) Exit?
[3/4/X]
```

Nous pouvons voir que les attributs du justificatif reçu correspondent à ceux défini par le schéma de Faber.

Connexion entre l'ACME et Alice

Nous établissons une connexion entre l'ACME et Alice. Nous commençons par démarrer l'agent de l'ACME sur un nouveau terminal.

```
# Terminal de l'ACME
valonrexhepi@WINDOWSDESKTOP:~/aries-cloudagent-python/demo$ LEDGER_URL=http://dev.greenlight.bcovrin.vonx.io ./run_demo acme
...
Acme        - DID: TdzzcSD1SSPt6jTuXgZ6Fi # DID public de l'ACME
...
Publish schema/cred def duration: 10.35s # publication des définitions sur le registre distribué
...
# Lien de l'invitation
Invitation Data:
{"@type": "https://didcomm.org/out-of-band/1.0/invitation", "@id": "95f6d53e-2272-407b-918e-110ef07c191b", "handshake_protocols": ["https://didcomm.org/didexchange/1.0"], "label": "acme.agent", "services": [{"id": "#inline", "type": "did-communication", "recipientKeys": ["did:key:z6Mkgp37e8PwkgjPp5a2JEM3GJHn21DzyXLJtTuDWUF5KRci"], "serviceEndpoint": "http://192.168.65.3:8040"}]}
...
Waiting for connection...
```

Le processus de lancement est le même que celui présenté dans la figure 4.14. Notons que le schéma développé dans le contrôleur *"acme.py"*, section 4.2.3, est bien inscrit dans le registre distribué, nous pouvons le vérifier au lien suivant.

Alice utilise l'option **[4]** sur son terminal pour copier le lien d'invitation de l'ACME.

```
# Terminal d'Alice
[3/4/X] 4
# En attente d'une invitation
Invite details: {"@type": "https://didcomm.org/out-of-band/1.0/invitation", "@id":
↪  "95f6d53e-2272-407b-918e-110ef07c191b",
↪  "handshake_protocols":["https://didcomm.org/didexchange/1.0"], "label": "acme.agent",
↪  "services": [{"id": "#inline", "type": "did-communication",
↪  "recipientKeys":["did:key:z6Mkgp37e8PwkgjPp5a2JEM3GJHn21DzyXLJtTuDWUF5KRci"],
↪  "serviceEndpoint": "http://192.168.65.3:8040"}]}
...
# Alice est connectée à l'ACME
...
    (3) Send Message
    (4) Input New Invitation
    (X) Exit?
[3/4/X]
```

Après la connexion, les terminaux de l'ACME et Alice affichent les options de communication disponibles.

```
# Terminal de l'ACME
Acme       | Connected
acme.agent handle_connections completed completed
    (1) Issue Credential
    (2) Send Proof Request
    (3) Send Message
    (X) Exit?
[1/2/3/X]
# ------------------------------------------------------------------------
# Terminal d'Alice
Alice      | Connected
Alice      | Check for endorser role ...
Connect duration: 0.18s
    (3) Send Message
    (4) Input New Invitation
    (X) Exit?
[3/4/X]
```

Notons que l'architecture de la connexion entre les deux agents est la même que sur la figure 4.16.

Échange d'une preuve entre Alice et l'ACME

Cette étape correspond à la relation **(2)** du triangle de confiance représenté par la figure 4.13.

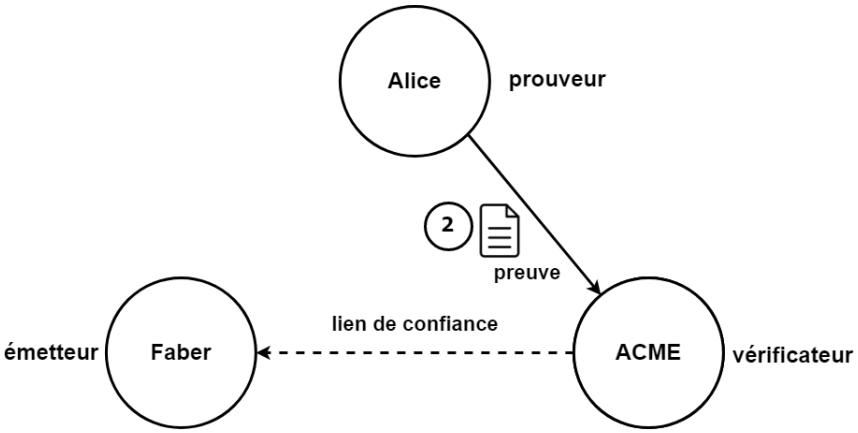


FIGURE 4.19 – *Deuxième relation du triangle de confiance*
Source: *de l'auteur*

Le schéma 4.20 représente le processus d'échange d'une preuve entre l'agent d'Alice et l'ACME.

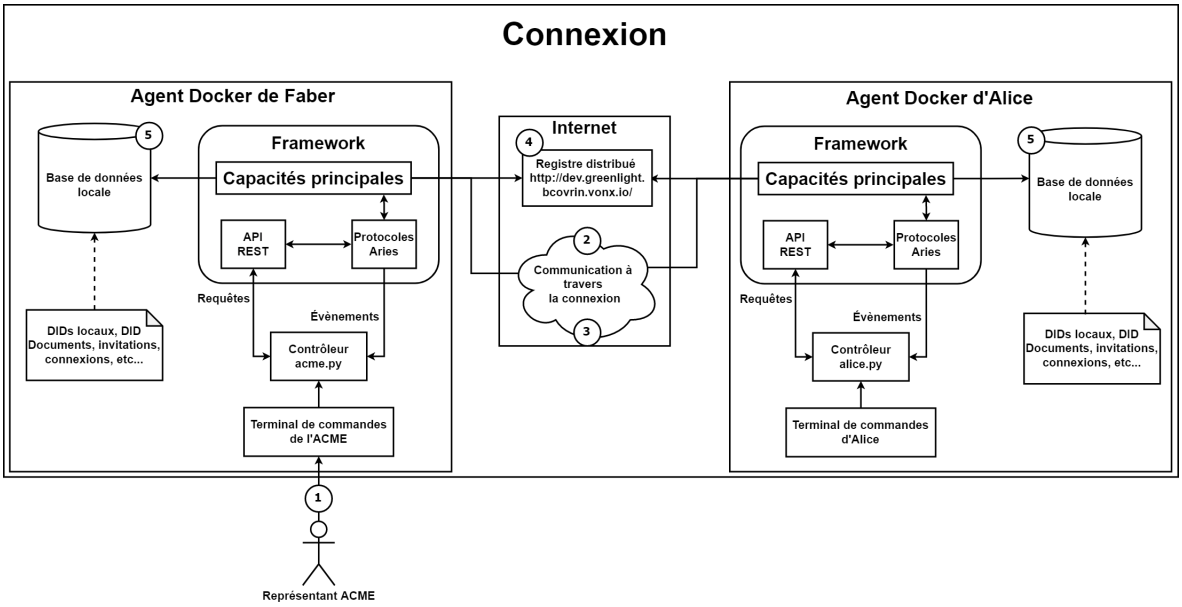


FIGURE 4.20 – *Processus d'échange d'une preuve entre l'agent d'Alice et l'ACME*
Source: *de l'auteur*

Pour rappel, nous avons développé les méthodes du contrôleur de l'ACME dans la section 4.2.3 et 4.2.3.

Pour initier l'échange, un représentant de l'entreprise ACME commence par interagir **(1)** avec le terminal en choisissant l'option **[2]**. Celle-ci envoie la demande de preuve à Alice.

```
# Terminal de l'ACME
[1/2/3/X]2

# Demande de preuve d'un diplôme à Alice
Acme       | Presentation: state = request-sent, pres_ex_id =
↪  5c52a217-a36f-40b9-95d5-6feb2d8a44ba
Acme       | Presentation: state = presentation-received, pres_ex_id =
↪  5c52a217-a36f-40b9-95d5-6feb2d8a44ba
```

La demande est envoyée **(2)** à l'agent d'Alice. Son contrôleur se charge automatiquement d'accepter et de répondre **(3)** à la demande.

```
# Terminal d'Alice
# Demande de preuve reçue
Alice      | Presentation: state = request-received, pres_ex_id =
↪  2cc0c6d8-2008-4887-be6c-2f5293258428

# Recherche d'une preuve dans le portefeuille qui satisfait la demande
# La preuve est générée
# La preuve est envoyée à l'ACME: {"indy": {"requested_predicates": {},
↪  "requested_attributes": {"0_name_uuid": {"cred_id":
↪  "7996c30e-6ff5-4206-b731-8786c2c743d1", "revealed": true}, "0_date_uuid": {"cred_id":
↪  "7996c30e-6ff5-4206-b731-8786c2c743d1", "revealed": true}, "0_degree_uuid": {"cred_id":
↪  "7996c30e-6ff5-4206-b731-8786c2c743d1", "revealed": true}}, "self_attested_attributes":
↪  {}}}
Alice      | Presentation: state = presentation-sent, pres_ex_id =
↪  2cc0c6d8-2008-4887-be6c-2f5293258428
Alice      | Presentation: state = done, pres_ex_id = 2cc0c6d8-2008-4887-be6c-2f5293258428
```

Le contrôleur de l'ACME reçoit la preuve d'Alice. Il va la contrôler et la valider. La validation s'effectue en vérifiant **(4)** la définition du schéma, définie par Faber, sur le registre distribué. Cela montre le lien de confiance entre l'ACME et Faber.

```
# Terminal de l'ACME
# Preuve d'Alice reçue
# Vérification de la validité de la preuve
Acme       | Presentation: state = done, pres_ex_id = 5c52a217-a36f-40b9-95d5-6feb2d8a44ba
Acme       | Proof =  true

# Vérification des attributs de la preuve
Acme       | name: Alice Smith
Acme       | date: 2018-05-28
Acme       | degree: Maths
Acme       | schema_id: AqRhLSCqnPcCpsW9W1bouU:2:degree schema:57.2.98
Acme       | cred_def_id AqRhLSCqnPcCpsW9W1bouU:3:CL:142837:faber.agent.degree_schema
```

Une trace de l'échange est inscrite **(5)** dans la base de données locale des deux agents.

Les terminaux affichent maintenant les options suivantes.

```
# Terminal de l'ACME
    (1) Issue Credential
    (2) Send Proof Request
    (3) Send Message
    (X) Exit?
[1/2/3/X]
# -------------------------------------------------------------------------
# Terminal d'Alice
    (3) Send Message
    (4) Input New Invitation
    (X) Exit?
[3/4/X]
```

Échange d'un justificatif entre l'ACME et Alice

Cette étape correspond à la relation **(3)** du triangle de confiance représneté par la figure par la figure 4.13.

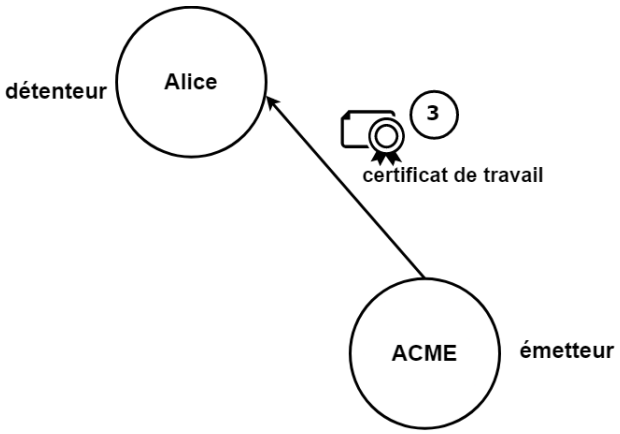


FIGURE 4.21 – *Troisième relation du triangle de confiance*
Source: *de l'auteur*

Ce processus est identique à celui présenté dans la figure 4.18. Le représentant de l'entreprise ACME va choisir l'option **[1]** sur son terminal pour envoyer un justificatif à Alice.

```
# Terminal de l'ACME
# Envoi de l'offre à Alice
Acme       | Credential: state = offer-sent, cred_ex_id =
↪  91ea9d50-763e-47b2-b026-6bed7e7d9834
Acme       | Credential: state = request-received, cred_ex_id =
↪  91ea9d50-763e-47b2-b026-6bed7e7d9834

# Envoi du justificatif à Alice
Acme       | Credential: state = credential-issued, cred_ex_id =
↪  91ea9d50-763e-47b2-b026-6bed7e7d9834
Acme       | Credential: state = done, cred_ex_id = 91ea9d50-763e-47b2-b026-6bed7e7d9834
    (1) Issue Credential
    (2) Send Proof Request
    (3) Send Message
    (X) Exit?
```

L'ACME envoie d'abord une offre de justificatif. Lorsque celle-ci est acceptée, le certificat de travail est transmis. Alice le stocke localement et affiche le détail du justificatif reçu sur son terminal.

```
# Terminal d'Alice
# Réception de l'offre de l'ACME
Alice      | Credential: state = offer-received, cred_ex_id =
↪  05c356f2-0da6-42d6-92a6-d9b6182b538f
Alice      | Credential: state = request-sent, cred_ex_id =
↪  05c356f2-0da6-42d6-92a6-d9b6182b538f

# Réception du certificat de travail
Alice      | Credential: state = credential-received, cred_ex_id =
↪  05c356f2-0da6-42d6-92a6-d9b6182b538f

# Stockage du justificatif
Credential details:
  {
    "referent": "3f6450ed-a756-4009-a6f4-7e6e9aaa21e9",
    "schema_id": "TdzzcSD1SSPt6jTuXgZ6Fi:2:employee id schema:82.37.28",
    "cred_def_id": "TdzzcSD1SSPt6jTuXgZ6Fi:3:CL:142862:acme.agent.employee_id_schema",
    "rev_reg_id": null,
    "cred_rev_id": null,
    "attrs": {
      "name": "Alice Smith",
      "employee_id": "ACME0009",
      "date": "2022-07-14",
      "position": "CEO"
    }
  }

Alice      | credential_id 3f6450ed-a756-4009-a6f4-7e6e9aaa21e9
Alice      | cred_def_id TdzzcSD1SSPt6jTuXgZ6Fi:3:CL:142862:acme.agent.employee_id_schema
Alice      | schema_id TdzzcSD1SSPt6jTuXgZ6Fi:2:employee id schema:82.37.28
Alice      | Credential: state = done, cred_ex_id = 05c356f2-0da6-42d6-92a6-d9b6182b538f
    (3) Send Message
    (4) Input New Invitation
    (X) Exit?
```

Nous pouvons constater que les attributs du justificatif reçu sont ceux définis par le schéma que nous avons développé dans le contrôleur de l'ACME.

Nettoyage

L'option **[x]** permet de stopper les terminaux de l'ACME et Alice.

```
# Terminal de l'ACME
[1/2/3/X]x
Shutting down agent ...
Acme       |
Acme       | Shutting down
Acme       | Exited with return code 0
# ------------------------------------------------------------------------
# Terminal d'Alice
Shutting down agent ...
Alice      |
Alice      | Shutting down
Alice      | Exited with return code 0
```

Nous pouvons ensuite utiliser les mêmes commandes[22] que dans le *PoC* précédent pour nettoyer le contenu de *Docker*.

```
valonrexhepi@WINDOWSDESKTOP:~$ docker stop $(docker ps -a -q)
valonrexhepi@WINDOWSDESKTOP:~$ docker rm $(docker ps -a -q)
valonrexhepi@WINDOWSDESKTOP:~$ docker rmi $(docker images -a -q)
valonrexhepi@WINDOWSDESKTOP:~$ docker system prune -a -f
```

4.2.5 Résultats

Dans ce *PoC*, nous avons commencé par établir certaines bases de connaissances sur le fonctionnement des contrôleurs. Avec ces notions, nous avons développé le contrôleur de l'entreprise ACME. Cela nous a permis d'en apprendre plus sur le ***framework*** *Cloud Agent Python*.

Nous avons ensuite lancé les différents agents préconfigurés d'Alice, Faber et de l'ACME. Ce lancement a généré, pour chacun, une interface en ligne de commandes que nous avons pu utiliser pour interagir avec l'agent et ainsi effectuer le scénario.

La réalisation de cette démonstration ne nécessite pas de connaissances poussées sur le fonctionnement du ***framework***. Cependant, il est nécessaire de passer du temps sur la lecture du code des contrôleurs pour bien les comprendre. Comme pour la première démonstration, la documentation pour la mise en place de ce second *PoC* est correctement maintenue par la communauté *Hyperledger*, ce qui permet une bonne compréhension générale des actions effectuées et de leur reproduction.

22. Pour rappel, ces commandes effacent tous les conteneurs, images et volumes.

5 | Conclusion

Dans ce chapitre, nous allons synthétiser les découvertes effectuées pendant la réalisation de ce document. Nous nous permettrons également de faire certaines recommandations sur la mise en place de l'écosystème de la *SSI*. Nous rappellerons ensuite les limites du travail et finalement nous ouvrirons la discussion sur des perspectives de recherches futures.

5.1 Synthèse

Dans ce travail, notre recherche a débuté par la compréhension des différents éléments qui composent l'écosystème de la *Self-Sovereign Identity*. Nous avons alors présenté l'identité autogérée, les registres distribués ainsi que quelques solutions existantes sur le marché. Ensuite, nous avons tenté de répondre, grâce aux connaissances acquises, aux différents défis identifiés par le *Département fédéral de justice et police*. Finalement, à l'aide de la solution *Hyperledger Aries*, nous avons mis en place différentes *Proof of Concept* pour démontrer la réalisation de différentes techniques pour l'intégration de la *SSI*.

5.2 Recommandations

Comme nous l'avons constaté tout au long de ce document, la *Self-Sovereign Identity* est une technologie prometteuse mais qui présente également des inconvénients. Il est important de ne pas se précipiter sur son adoption car de nombreuses questions sont encore sans réponses. De plus, certaines technologies qui la composent doivent être attentivement choisies pour ne pas représenter de failles potentielles à sa mise en place. Nous avons vu, par exemple, qu'un *Distributed Ledger* peut consommer trop d'énergie et représenter alors un danger pour la planète.

En ce qui concerne les solutions existantes du moment, la plus mature et la plus simple à prendre en main est *MATTR VII*, mais celle-ci à des inconvénients qui limitent son utilisation pour la mise en place d'un ***e-ID***. La solution *Aries* proposée par la fondation *Hyperledger* reste néanmoins un excellent choix pour la création de *PoC*, mais est pour le moment encore trop incomplète pour proposer une solution adaptable à tous les cas d'utilisations.

Enfin, ce manque de maturité se retrouve également dans d'autres technologies qui composent la *SSI*, confirmant de ce fait l'importance de ne pas se précipiter dans son intégration.

5.3 Limites du travail

Nous rappelons que ce projet a été réalisé dans la période du 5 mai au 29 juillet 2022 dans un cadre scolaire. Les ressources à notre disposition sont de ce fait limitées, que cela soit au niveau temporel ou technique.

5.4 Perspectives de recherches

Le 29 juin 2022, une nouvelle proposition d'initiative sur la mise en place de l'***e-ID*** a été mise en consultation par le Conseil fédéral. C'est bien l'identité autogérée qui est retenue pour la mise en place de celui-ci. Cependant, il reste de nombreuses questions philosophiques et techniques, autour de cette technologie, en suspens.

Quelles technologies vont être utilisées pour mettre en place cette *SSI* ? Est-ce que l'état aura un contrôle sur cet identifiant électronique ? Sera-il possible de voyager grâce à son ***e-ID*** ? Est-ce que celui-ci pourra être utilisé en dehors de la Suisse ?

Ce ne sont là qu'un échantillon des nombreuses questions qui entourent cette technologie, mais d'une façon plus générale, nous pouvons nous demander à quel point la mise en place de cet ***e-ID*** va influencer la vie des citoyens suisses.

I Annexe - Journal de bord

Dans cette annexe, nous pouvons voir le journal de bord effectué pendant la durée de ce projet. Chaque entrée est composée de la journée et de la date de travail, d'une rapide vue d'ensemble de l'activité réalisée durant cette journée, le temps de travail investi ainsi que les informations sur les documents associés.

La colonne *"Document Associé"* indique le nom d'un rapport rempli expliquant en détail les objectifs et le travail réalisés à la date mentionnée. La colonne *"Labs"* contient les noms des documents, de démonstrations et de recherches techniques, effectués. Ces *"labs"* sont trouvables dans l'annexe suivante.

Journée	Date	Activité	Temps Travail	Document Associé	Labs
Jeudi Matin	05.05.2022	Réunion et commencement du projet	-	-	-
Lundi	09.05.2022	Lecture SSI	08 h 00	-	-
Mardi	10.05.2022	Lecture SSI	08 h 00	-	-
Mercredi Matin	11.05.2022	Lecture SSI	02 h 00	-	-
Jeudi Matin	12.05.2022	Réunion avec les membres du groupes SSI, premières pistes d'explorations	03 h 00	-	-
Vendredi	13.05.2022	Test d'implémentation en Go et Python, comparaison des deux, feedbacks à l'équipe	08 h 00	-	-
Samedi	14.05.2022	Test d'implémentation en Go et Python, comparaison des deux, feedbacks à l'équipe	08 h 00	-	-
Lundi	16.05.2022	Lecture SSI, meeting, recherche ACA-Py, suivi des cours EDX, réalisation de Labs	13 h 45	Suivi_16_05_2022	Lab1
Mardi	17.05.2022	Exploration Aries, suivi des cours EDX, réalisation de Labs	08 h 00	Suivi_17_05_2022	Lab2, Lab3, Lab4
Mecredi	18.05.2022	Exploration Aries et réalisation de labs	05 h 00	Suivi_18_05_2022	Lab5, Lab6
Jeudi Matin	19.05.2022	Commencement de la rédaction du bachelor	03 h 00	Suivi_19_05_2022	
Lundi	23.05.2022	Exploration Aries, réalisation de labs et meeting	12 h 00	Suivi_23_05_2022	Lab7
Mardi	24.05.2022	Réalisation de labs et rédaction du bachelor	08 h 00	Suivi_24_05_2022	Lab7
Mercredi Matin	25.05.2022	Rédaction du bachelor	04 h 00	Suivi_25_05_2022	-
Jeudi	26.05.2022	Rédaction du bachelor	03 h 00	Suivi_26_05_2022	-
Vendredi	27.05.2022	Rédaction du bachelor	07 h 30	Suivi_27_05_2022	-
Samedi	28.05.2022	Rédaction du bachelor	01 h 00	Suivi_28_05_2022	-
Lundi	30.05.2022	Rédaction du bachelor et meeting	08 h 30	Suivi_30_05_2022	-
Mardi	31.05.2022	Rédaction du bachelor	11 h 00	Suivi_31_05_2022	-
Mercredi Matin	01.06.2022	Recherche Architecture HyperLedger	01 h 30	Suivi_01_06_2022	-
Jeudi Matin	02.06.2022	Recherche Architecture HyperLedger	03 h 00	Suivi_02_06_2022	Lab8
Samedi	04.06.2022	Recherche Architecture HyperLedger	04 h 00	Suivi_04_06_2022	Lab8
Lundi	06.06.2022	Rédaction du bachelor	09 h 30	Suivi_06_06_2022	-
Mardi	07.06.2022	Rédaction du bachelor	09 h 00	Suivi_07_06_2022	-
Mercredi	08.06.2022	Rédaction du bachelor	09 h 00	Suivi_08_06_2022	-
Jeudi Matin	09.06.2022	Rédaction du bachelor	03 h 00	Suivi_09_06_2022	-
Samed Matin	11.06.2022	Rédaction du bachelor	05 h 00	Suivi_11_06_2022	-
Lundi	13.06.2022	Rédaction du bachelor et meeting	08 h 00	Suivi_13_06_2022	-
Mardi	14.06.2022	Rédaction du bachelor	07 h 00	Suivi_14_06_2022	-
Mercredi	15.06.2022	Rédaction du bachelor	03 h 00	Suivi_15_06_2022	-
Lundi	20.06.2022	Séance coaching 180 secondes et meeting	03 h 00	Suivi_20_06_2022	-
Lundi	27.06.2022	Rédaction du bachelor et meeting	10 h 30	Suivi_26_06_2022	-
Mardi	28.06.2022	Rédaction du bachelor et meeting	10 h 00	Suivi_27_06_2022	-
Mercredi	29.06.2022	Rédaction du bachelor	11 h 00	Suivi_28_06_2022	-
Jeudi	30.06.2022	Rédaction du bachelor	05 h 00	Suivi_30_06_2022	-
Vendredi	01.07.2022	Rédaction du bachelor	05 h 00	Suivi_01_07_2022	-
Samedi	02.07.2022	Rédaction du bachelor et préparation du poster	03 h 30	Suivi_02_07_2022	-
Lundi	04.07.2022	Rédaction du bachelor et Meeting	10 h 00	Suivi_04_07_2022	-
Mardi	05.07.2022	Rédaction du bachelor	14 h 00	Suivi_05_07_2022	-
Mercredi	06.07.2022	Rédaction du bachelor	06 h 00	Suivi_06_07_2022	-
Jeudi	07.07.2022	Recherche technologies SSI et suivi conférence	08 h 30	Suivi_07_07_2022	-
Vendredi	08.07.2022	Recherche technologies SSI	04 h 00	Suivi_08_07_2022	-
Samedi	09.07.2022	Recherche technologies SSI	03 h 45	Suivi_09_07_2022	-
Lundi	11.07.2022	Recherche technologies SSI et rédaction du bachelor	13 h 30	Suivi_11_07_2022	-
Mardi	12.07.2022	Réunion et correction du bachelor	08 h 30	Suivi_12_07_2022	-
Mercredi	13.07.2022	Rédaction du bachelor	16 h 00	Suivi_13_07_2022	-
Jeudi	14.07.2022	Rédaction du bachelor	13 h 00	Suivi_14_07_2022	-
Vendredi	15.07.2022	Rédaction du bachelor	10 h 00	Suivi_15_07_2022	-
Samedi	16.07.2022	Relecture du bachelor	06 h 00	Suivi_16_07_2022	-
Lundi	18.07.2022	Corrections finales, relecture et préparation au rendu final	07 h 30	Suivi_17_07_2022	-
				Total actuel	352 h 30

II | Annexe - Aries Labs

Dans cette annexe, nous pouvons retrouver toutes les démonstrations effectuées durant la durée de ce projet. Celles-ci se présentent sous la forme d'exercices nommés *labs*. Ils sont réalisés au départ avec le langage *markdown* mais sont inclus ici en tant que fichiers *pdf* générés.

Hyperledger Aries - Lab Utilisation Von Network et Création d'un DID

Ce document montre la reproduction du lab suivant via une machine wsl2 Ubuntu sur Windows 11.

Prérequis

Pour effectuer ce Lab, il est nécessaire d'avoir :

- Un Browser
- Docker
- Git

Étapes de réalisation

Démarrer VON Network et accéder à l'interface Web

La première étape va être de cloner le repository de VON Network :

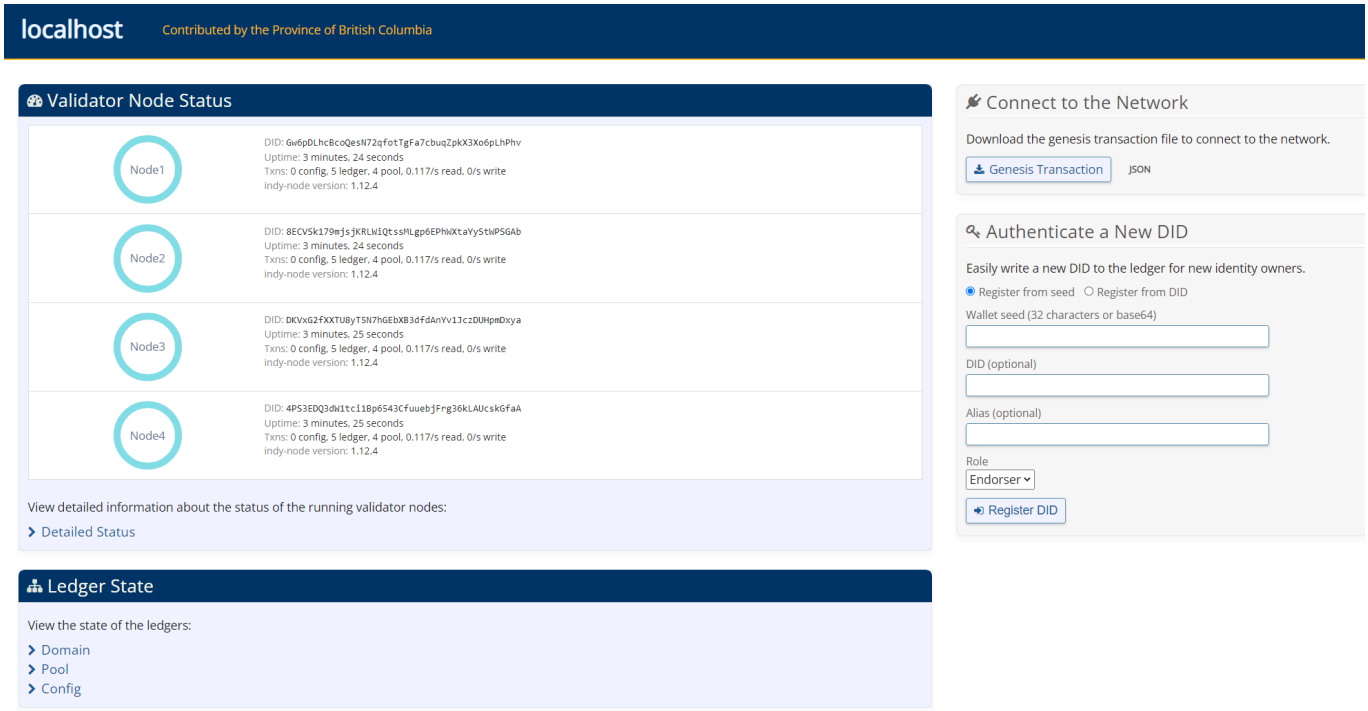
```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython$ git clone https://github.com/bcgov/von-
network
Cloning into 'von-network'...
remote: Enumerating objects: 1918, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 1918 (delta 0), reused 0 (delta 0), pack-reused 1914
Receiving objects: 100% (1918/1918), 58.37 MiB | 22.30 MiB/s, done.
Resolving deltas: 100% (1102/1102), done.
valonrexhepi@WINDOWSDESKTOP:~/AriesPython$ cd von-network/
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/von-network$
```

Ensuite, nous allons "build" le VON network et le démarrer en affichant les différents logs :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/von-network$ ./manage build
...
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/von-network$ ./manage start --logs
...
von-webserver-1  | 2022-05-16 19:30:40,150|DEBUG|anchor.py|Finished resync
```

Nous pouvons à tout moment faire un CTRL + C dans la console pour quitter les logs, cela n'arrête pas le VON network. La commande "./manage logs" permet de retourner dans les logs du network.

Pour explorer le ledger, il suffit alors d'aller sur l'adresse http://localhost:9000 :

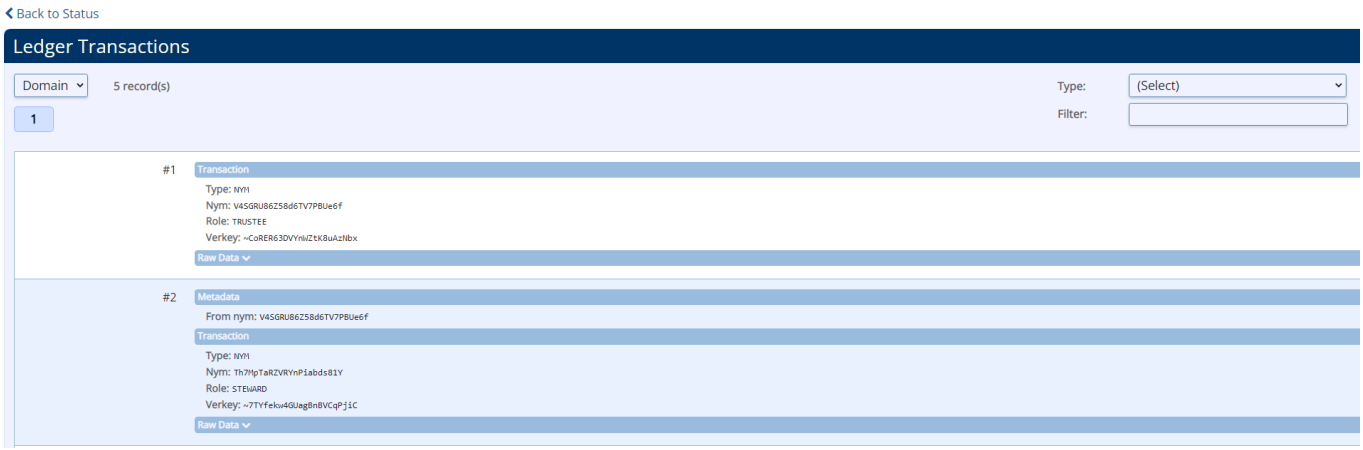


Depuis l'interface Web, il est possible de :

- voir le statut des nœuds sur le réseau
- avoir un aperçu du fichier genesis sur le réseau
- créer un DID
- parcourir les trois ledgers qui vont un réseau Indy :
- le Domain Ledger, où se trouvent les DIDs, schémas, etc...
- le Pool Ledger, où le set de nœuds sur le réseau sont traqués
- le Config Ledger, où les changements au réseau sont traqués

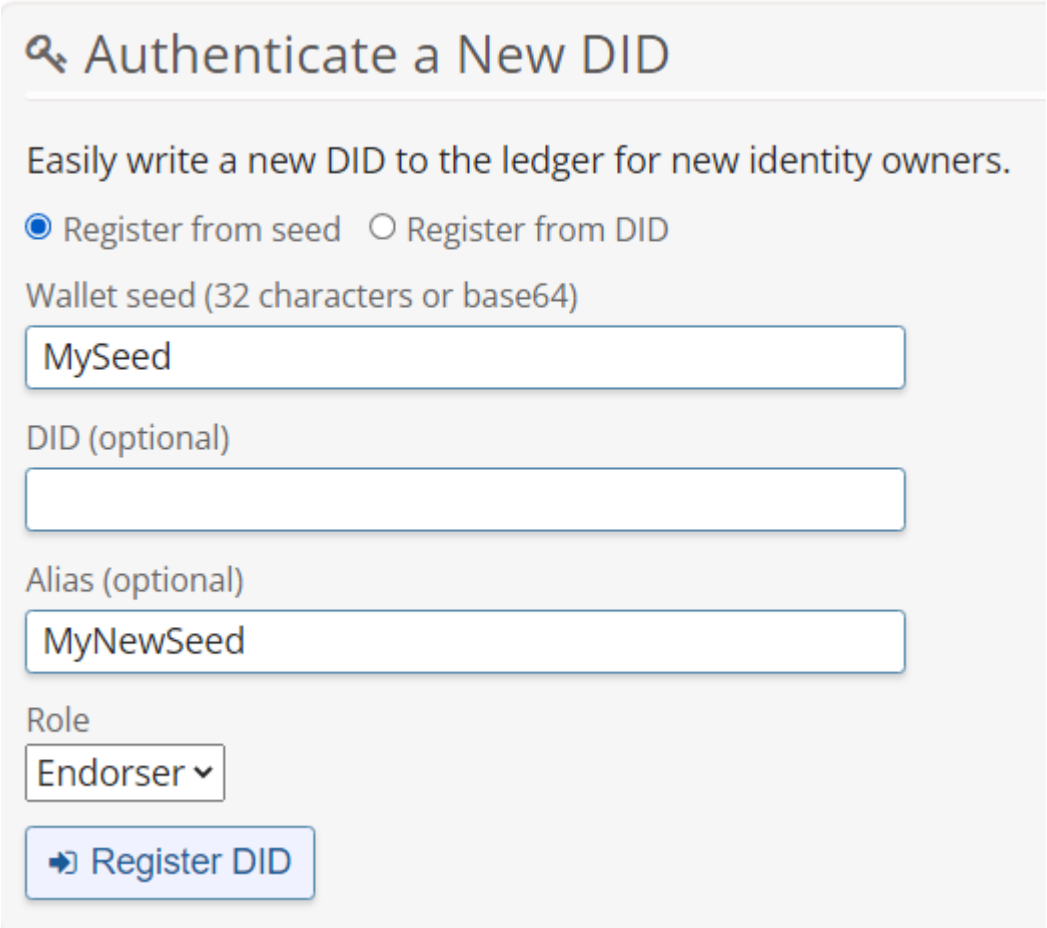
Voir des transactions

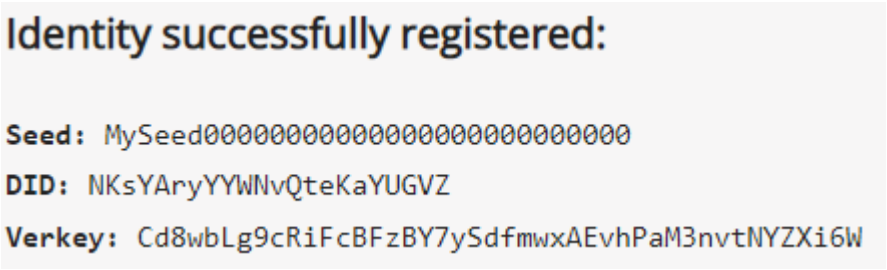
En cliquant sur le bouton "Domain" dans la rubrique Ledger State, nous pouvons voir les différentes transactions disponibles et les filtrer.



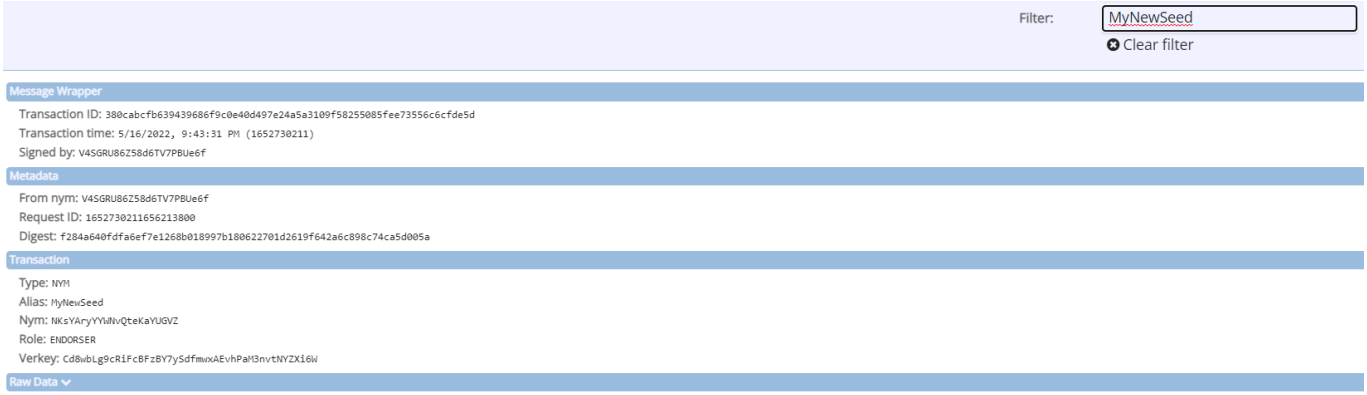
Créer un DID et le voir dans les transactions

Sur la page de statut, la page de départ, sur la droite, nous pouvons "Authenticate a New DID". Voilà un exemple de création :





Si nous allons dans les transactions maintenant, nous pouvons filtrer avec l'alias choisi par exemple ici "MyNewSeed".



Voir le fichier Genesis

Depuis la page d'accueil, en cliquant sur le bouton "Genesis Transaction", il est possible d'avoir un aperçu du fichier Genesis.


```
{"reqSignature":{},"txn":{"data":{"data":
{"alias":"Node1","blskey":"4N8aUNHSgjQVgkpm8nhNEfDf6txHznoYREg9kirmJrkivgL4oSEimFF
6nsQ6M41QvhM2Z33nves5vfSn9n1UwNFJBYtWVnHYMATn76vLuL3zU88KyeAYcHfsih3He6UHcXDxcaecH
Vz6jhCYz1P2UZn2bDVruL5wXpehgBfBaLKm3Ba","blskey_pop":"RahHYiCvoNCtPTrVtP7nMC5eTYrs
UA8WjXbdhNc8debh1agE9bGiJxWBXYNFbnJXoXhWFMvyqhqhRoq737YQemH5ik9oL7R4NTTCz2LEZhkgLJ
zB3QRQqJyBNyv7acbdHrAT8nQ9UkLbaVL9NBpnWXBTw4LEMePaSHEw66RzPNdAX1","client_ip":"192
.168.65.3","client_port":9702,"node_ip":"192.168.65.3","node_port":9701,"services"
:["VALIDATOR"]},"dest":"Gw6pDLhcBcoQesN72qfotTgFa7cbuqZpkX3Xo6pLhPhv"},"metadata":
{"from":"Th7MpTaRZVRYnPiabds81Y"},"type":"0"},"txnMetadata":
{"seqNo":1,"txnId":"fea82e10e894419fe2bea7d96296a6d46f50f93f9eeda954ec461b2ed2950b
62"},"ver":"1"}
{"reqSignature":{},"txn":{"data":{"data":
{"alias":"Node2","blskey":"37rAPpXVoxzKhz7d9gkUe52XuXryuLXoM6P6LbWDB7LSbG62Lsb33sf
G7zqS8TK1MXwuCHj1FKNzVpsnafmqLG1vXN88rt38mNFs9TENzm4QHdBzsvCuoBnPH7rpYYDo9DZNJePaD
vRvqJKByCabubJz3XXKbEeshzpz4Ma5QYpJqjk","blskey_pop":"Qr658mWZ2YC8JXGXwMDQTzuZCWF7
NK9EwxphGmcBvCh6ybUuLxbG65nsX4JvD4SPNtkJ2w9ug1yLTj6fgmuDg41TgECXjLCij3RMsV8CwewBVg
VN67wsA45DFWvqvLtu4rjNnE9JbdFTc1Z4WCPA3Xan44K1HoHAq9EVeaRYs8zoF5","client_ip":"192
.168.65.3","client_port":9704,"node_ip":"192.168.65.3","node_port":9703,"services"
:["VALIDATOR"]},"dest":"8ECVSk179mjsjKRLWiQtssMLgp6EPhWXtaYyStWPSGAb"},"metadata":
{"from":"EbP4aYNeTHL6q385GuVpRV"},"type":"0"},"txnMetadata":
{"seqNo":2,"txnId":"1ac8aece2a18ced660fef8694b61aac3af08ba875ce3026a160acbc3a3af35
fc"},"ver":"1"}
{"reqSignature":{},"txn":{"data":{"data":
{"alias":"Node3","blskey":"3WFpdbg7C5cnLYZwFZevJqhubkFALBfCBBok15GdrKMUhUjGsk3jV6Q
Kj6MZgEubF7oqCafxNdkm7eswgA4sdKTRc82tLGzZBd6vNqU8dupzup6uYUf32KTHTPQbuUM8Yk4QFXjEf
2Usu2TJcNkdgpyeUSX42u5LqdDDpNSWUK5deC5","blskey_pop":"QwDeb2CkNSx6r8QC8vGQK3GRv7Yn
dn84TGNijX8YXHPiagXajyfTjoR87rXUu4G4QLk2cF8NNyqWiYMus1623dELWwx57rLCFqGh7N4ZRbGDRP
4fnVcaKg1BcUxQ866Ven4gw8y4N56S5HzxXNBZtLYmhGHvDtk6PFkFwCvxYrNYjh","client_ip":"192
.168.65.3","client_port":9706,"node_ip":"192.168.65.3","node_port":9705,"services"
:["VALIDATOR"]},"dest":"DKVxG2fXXTU8yT5N7hGEbXB3dfdAnYv1JczDUHpmDxya"},"metadata":
{"from":"4cU41vWW82ArfxJxHkzXPG"},"type":"0"},"txnMetadata":
{"seqNo":3,"txnId":"7e9f355dffa78ed24668f0e0e369fd8c224076571c51e2ea8be5f26479edeb
e4"},"ver":"1"}
{"reqSignature":{},"txn":{"data":{"data":
{"alias":"Node4","blskey":"2zN3bHM1m4rLz54MJHYSwvqzPchYp8jkHswveCLAEJVcX6Mm1wHQD1S
kPYMzUDTZvWvhuE6VNAkK3KxVeEmsanSmvjVkReDeBEMxeDaayjcZjFGPydyey1qxBHmTvAnBKoPydvuTA
qx5f7YNNRAdeLmUi99gERUU7TD8KfAa6MpQ9bw","blskey_pop":"RPLagxaR5xdimFzwmzYnz4ZhWtYQ
Ej8iR5ZU53T2gitPCyCHQneUn2Huc4oeLd2B2HzkGnjAff4hWTJT6C7qHYB1Mv2wU5iHHGFWkhnTX9WsEA
bunJCV2qcaXScKj4tTfvdDKfLiVuU2av6hbsMztirRze7LvYBkRHV3tGwyCptsrP","client_ip":"192
.168.65.3","client_port":9708,"node_ip":"192.168.65.3","node_port":9707,"services"
:["VALIDATOR"]},"dest":"4PS3EDQ3dW1tci1Bp6543CfuuebjFrg36kLAUcskGfaA"},"metadata":
{"from":"TWwCRQRZ2ZHMJFn9TzLp7W"},"type":"0"},"txnMetadata":
{"seqNo":4,"txnId":"aa5e817d7cc626170eca175822029339a444eb0ee8f0bd20d3b0b76e566fb0
08"},"ver":"1"}
```

Comme nous l'indique le tutoriel, quand on exécute une instance de VON Network avec le Ledger Browser, il est possible d'obtenir le fichier genesis via le endpoint /genesis, ainsi ceci peut être utilisé en tant que paramètre d'un agent Aries.

Obtenir une CLI pour Indy

Pour lancer une interface de ligne de commande pour Indy, il faut retourner sur le dossier de von-network et taper ceci :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/von-network$ ./manage indy-cli

Using: docker --log-level error compose

indy>
```

Il est ensuite possible de tapper help pour voir les différentes commandes disponibles par exemple :

```
indy> help
Hyperledger Indy CLI

Usage:
        [<command-group>] <command> [[<main-param-name>=]<main-param-value>]
[<param_name-1>=<param_value-1>]...[<param_name-n>=<param_value-n>]

Getting help:
        help - Display this help
        <command-group> help - Display the help for the specific command group
        [<command-group>] <command> help - Display the help for the specific
command

Command groups are:
        ledger - Ledger management commands
        pool - Pool management commands
        did - Identity management commands
        payment-address - Payment address management commands
        wallet - Wallet management commands

Top level commands are:
        prompt - Change command prompt
        exit - Exit Indy CLI
        init-logger - Init logger according to a config file.
        Indy Cli uses `log4rs` logging framework: https://crates.io/crates/log4rs
        about - Show about information
        show - Print the content of text file
        load-plugin - Load plugin in Libindy

indy> did help
Group:
        did - Identity management commands
```

```
Usage:
        did <command> [[<main-param-name>=]<main-param-value>] [<param_name-1>=
<param_value-1>]...[<param_name-n>=<param_value-n>]

Getting help:
        did <command> help - Display the help for the specific command

Group commands are:
        use - Use DID
        rotate-key - Rotate keys for active did
        list - List my DIDs stored in the opened wallet.
        qualify - Update DID stored in the wallet to make fully qualified, or to
do other DID maintenance.
        new - Create new DID
        import - Import DIDs entities from file to the current wallet.
        File format:
        {
            "version": 1,
            "dids": [{
                "did": "did",
                "seed": "UTF-8, base64 or hex string"
            }]
        }

indy>
```

Stopper et arrêter VON Network

Nous pouvons stopper le service le VON network sans supprimer son contenu et le redémarrer autant de fois que cela nous plaît comme ceci :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/von-network$ ./manage stop
...
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/von-network$ ./manage start
...
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/von-network$
```

Pour stopper totalement le VON network et le supprimer, nous utilisons le paramètre "down" comme ceci :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/von-network$ ./manage down
...
```

Hyperledger Aries - Lab Utilisation d'un Universal Resolver

Ce document montre la reproduction du lab suivant via le site internet Universal Resolver.

Prérequis

Pour effectuer ce Lab, il est nécessaire d'avoir :

- un navigateur internet

Avant-Propos sur l'utilisation d'un Universal Resolver

Hyperledger Aries a débuté en utilisant que l'Indy Ledgers et les Indy DIDs, mais aujourd'hui il peut gérer plusieurs types de DID grâce à l'utilisation d'une instance d'Universal Resolver.

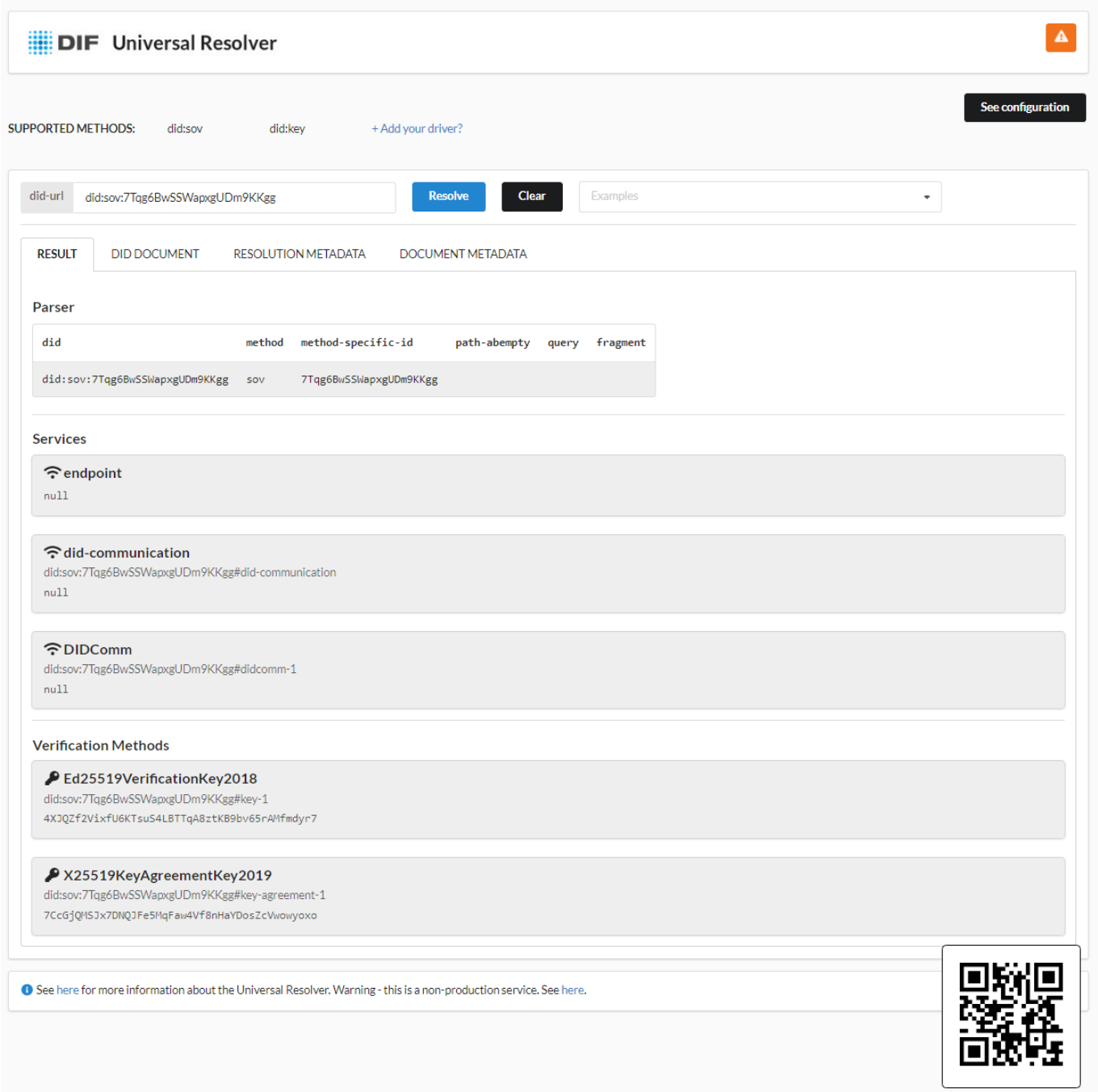
Il est fortement recommandé de ne jamais compter sur une instance centralisée publique comme l'Universal Resolver que pour des tâches autres que des démos non-critiques ou du développement.

Il est conseillé de créer sa propre instance d'Universal Resolve ou d'autres outils de résolution de DID et de l'intégrer à son application.

Étapes de réalisation

Exemple de résolution d'un did:sov

Nous allons utiliser un DID trouvé sur la Sovrin MainNet comme premier exemple de résolution de DID. Nous allons donc copier le DID suivant "did:sov:7Tqg6BwSSWapxgUDm9KKgg" dans le did-url et cliquer sur Resolve pour voir le résultat.



Dans l'ordre, les onglets [DID DOCUMENT, RESOLUTION METADATA, DOCUMENT METADATA] comportent les informations suivantes :


```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2018/v1",
    "https://w3id.org/security/suites/x25519-2019/v1"
  ],
  "id": "did:sov:7Tqg6BwSSWapxgUDm9KKgg",
  "verificationMethod": [
    {
      "type": "Ed25519VerificationKey2018",
      "id": "did:sov:7Tqg6BwSSWapxgUDm9KKgg#key-1",
      "controller": "did:sov:7Tqg6BwSSWapxgUDm9KKgg",
```

```
        "publicKeyBase58": "4XJQZf2VixfU6KTsuS4LBTTqA8ztKB9bv65rAMfmdyr7"
      },
      {
        "type": "X25519KeyAgreementKey2019",
        "id": "did:sov:7Tqg6BwSSWapxgUDm9KKgg#key-agreement-1",
        "controller": "did:sov:7Tqg6BwSSWapxgUDm9KKgg",
        "publicKeyBase58": "7CcGjQMSJx7DNQJFe5MqFaw4Vf8nHaYDosZcVwowyoxo"
      }
    ],
    "authentication": [
      "did:sov:7Tqg6BwSSWapxgUDm9KKgg#key-1"
    ],
    "assertionMethod": [
      "did:sov:7Tqg6BwSSWapxgUDm9KKgg#key-1"
    ],
    "keyAgreement": [
      "did:sov:7Tqg6BwSSWapxgUDm9KKgg#key-agreement-1"
    ],
    "service": [
      {
        "type": "endpoint",
        "serviceEndpoint": "null"
      },
      {
        "type": "did-communication",
        "id": "did:sov:7Tqg6BwSSWapxgUDm9KKgg#did-communication",
        "serviceEndpoint": "null",
        "recipientKeys": [
          "did:sov:7Tqg6BwSSWapxgUDm9KKgg#key-1"
        ],
        "priority": 0,
        "routingKeys": [],
        "accept": [
          "didcomm/aip2;env=rfc19"
        ]
      },
      {
        "type": "DIDComm",
        "id": "did:sov:7Tqg6BwSSWapxgUDm9KKgg#didcomm-1",
        "serviceEndpoint": "null",
        "routingKeys": [],
        "accept": [
          "didcomm/v2",
          "didcomm/aip2;env=rfc19"
        ]
      }
    ]
  }
```



```
  {
    "contentType": "application/did+ld+json",
```

```
  "pattern": "^(did:sov:(?:(?:\\w[-\\w]*(?::\\w[-\\w]*)*):)?(?:[1-9A-HJ-NP-Za-km-
z]{21,22}))$",
  "driverUrl": "http://driver-did-sov:8080/1.0/identifiers/",
  "duration": 444,
  "did": {
    "didString": "did:sov:7Tqg6BwSSWapxgUDm9KKgg",
    "methodSpecificId": "7Tqg6BwSSWapxgUDm9KKgg",
    "method": "sov"
  }
}
```



```
{
  "network": "_",
  "poolVersion": 2,
  "submitterDid": "5Y9j3N2CB3Xm5GwbHaQd8m",
  "nymResponse": {
    "result": {
      "identifier": "5Y9j3N2CB3Xm5GwbHaQd8m",
      "state_proof": {
        "proof_nodes":
"+QV6+QIRoFYQqK+RHoVy8R0gTK2AZkKL8jCFqaY5hEKlfKdcL7+doLhvsNl5Lh0G7x9pttkUcnlIaaem8
xgo4wW/U16MLYggoE+XpokW0OT1mdRPFX3oW2utvnWI5t4QscxQDQJgI7kJoNgVsiZE/JxoolDgxkHl7SK
8jm4c3NFaapkAsCxE8+DAoJf/bTJIFsejLTQ5Ep01TRQnh7H9x3Ci5xkSoihwaCyDoEZatZeUWClFrOHIp
MoqegBa3Mx2xFdhL4thw7d2AVd6oOm1VpHnWZ2tp1vazKszPmhYxhvlvws/+ySsOWd8VzizoGL7qUOoFT7
ibAO42+mdYswxNZoU+B07zPrMtKfg5xLMoATK/xznYNYtbs6QMiWkPaP1VH255L+lMMES6B0g9bV7oDPRV
sULeQjFBekpfRcQ+/uY7ZQ1/4R3t0CnDQFoeSjEoJH4rX41w0T6kWNVUjcmArFOE/4cgjK5AgOVoSJa/dV
WoKc9EQB3z3tljri/Gy8arv4f2cE69fJ5DpGBP0a5SS15oCEebPRA0p0h8vP5gVA7rRqdRiWhEO7k9CbuF
LnZyw7CoBPpfUev9nLLKPZcu6YoxVsecjTA5SkUaGg/d60rnLsioFrFO+be9Yqxny6oVXwq9GY3c5f5Uij
HqT7nwNKTdDxmoHthXVAm1Tq5ZQW+8+BcZO0m8VVyoR3Lq6uGyMlj3skmgPixgICgKLuRfDNB0Fm5FGQNG
9C9UAiO4HkR2pUEQiSNGByKtAGAgKD85A+FgGvtKR5kMjlc9be8SmW0OfAiWTAeXCfhcujQjoCAgICAoN4
tGXZAwdfRlL65owgUFiMQF3erpLUZ1wEyOrNbfI7rgKCmPOovrn+ktC6bCejUsIxUdSlRfaG4hle/lLTE9
vdmHKD7ohLtMsLrq+G48pge/AfSbVwbFlz52wfjsx3kz76j0YCA+J2fPwP3vMyHpHlmcEUwA9PUF5Vy7/J
93+3geRSpaA61zLh7+Hm4d3siaWRlbnRpZmllciI6Im1ndHFHRUY4RUJRSEJMUkVkUm9KNCIsInJvbGUiO
iIwIiwic2VxTm8iOjU0NDc0LCJ0eG5UaW1lIjoxNTkyOTY3NTU3LCJ2ZXJrZXkiOiJ+QW1KeE13Y21RTXB
ldWdSNWdob1U4byJ9+QIRoL8TLgzZ00SLH9NRHFM/anNeLBqz/fmBboAoUz5WO9IPoCs6HPv6ffuO3hg/u
J5GzXl/oGZD/NZYw+Gpj6mJ6R8BoCLhRb9fXDi1ltXY4zBRaz83mpU0y4osDpl2CwwVkExToOu7dJB1e0V
BJhxxEAWFKfmtdesOgUgTha2iL3jWSrRcoO+Ls4G8OSS2qPzkub/luvxdsyZrBMdtl0aMfvdTM6/eoBYVh
n20J3EXFyirQzScwGYkbzZ9fd6zm5GDkqTpAh3FoBkMsXOPXmhssjnuKFTi+PoQBBZv+mshGAYEGEGH1DC
OoDBkadAXdb/VAwDIQxGSJB4jcddU609Cpy0Wbwa/+K2IoAOm3RyTLAmNpU7c7F/YnecF8zoQUJyJ/G4BT
xKvsbUYoA6VSh/YlPX/OWT47cKwAEbM17XpSg62EJQeMSl+cKvkoAdKxOl26bOTDXCMJR8GNRs1QM8qdWP
Oa0Am8V5EPQmFoPnn2lCvyqtWS6DO8pTAlkHu8CHUwiTf5+q1w27vfE9NoKyAmiCNbF0SWzVcfZ6zY3uye
ZmehrohiNVhrDSNf1KooE9QHZAXk5Gp068GVXQPEssb4AjtTELJGrDuwJaLYMmEoOEL1nfXQxp3UgqhiD9
gyS/ElFGXKR0JIw/HCOw4ENUgoN3wKRijBZNDfy2HbNCWcua4HcnXN74pHiDHl64EuMPfgA==",
        "root_hash": "Dcda5BCQSGiQfCGWEhXhtm3zkum4YABTGYRrNueaWhHX",
        "multi_signature": {
          "signature":
"RbCSjQEpzpac44mXdpn16kBQyBkFPhiSzCaykUhtQvhKtcir389UNBzJYkUMUFQAdrrXiJmampzuEpjnK
vLRiNgC6R6jtTXQuL1f6omH235wbnj79yNS2gW5VgNSXYnKE34CiMnAdBs1hkzsA7TrohZGbcXmnLAfphq
yy48Gkz5H2w",
          "participants": [
            "esatus_AG",
```

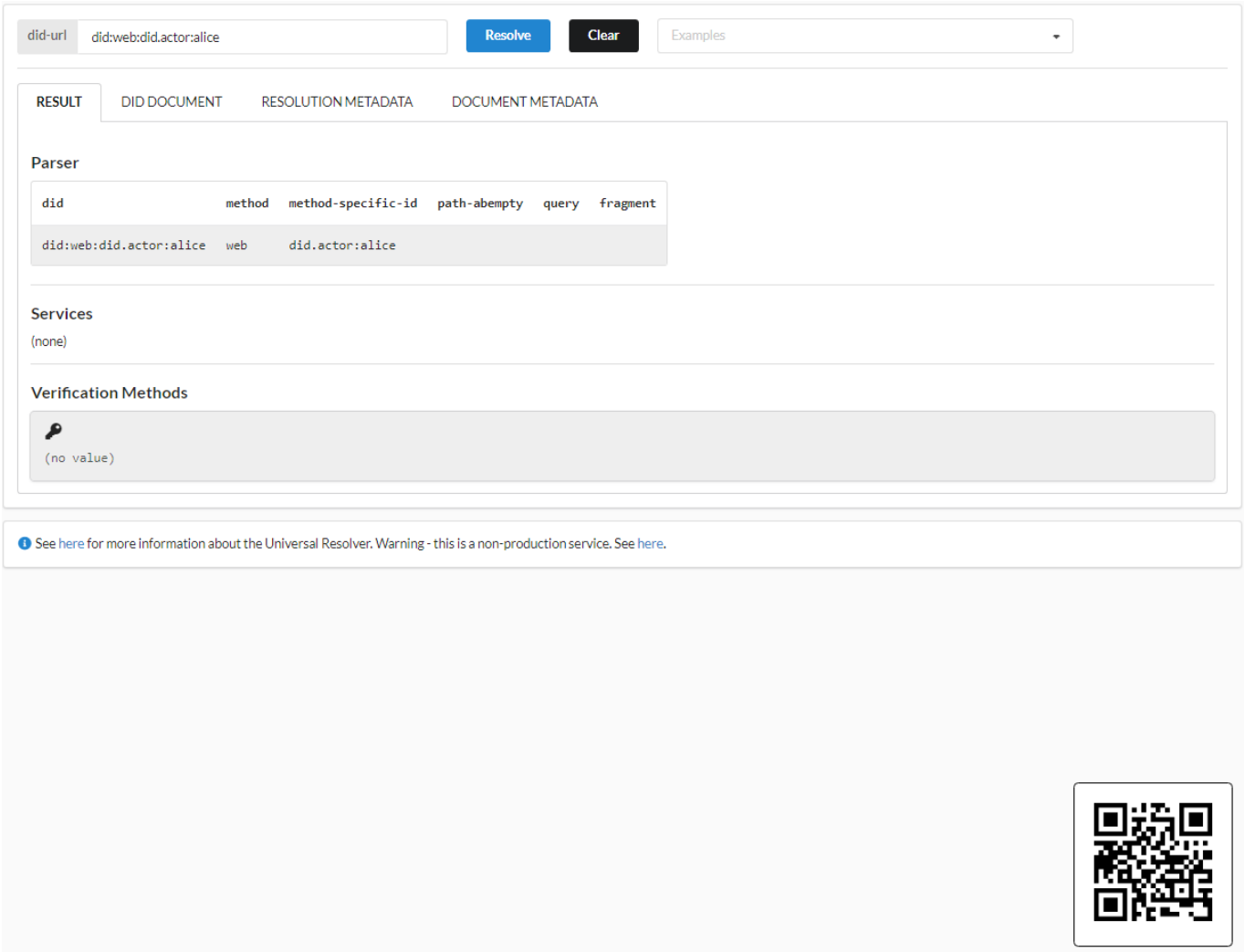
```
            "atbsovrin",
            "royal_sovrin",
            "OASFCU",
            "prosovitor",
            "VeridiumIDC",
            "danube",
            "sovrin.sicpa.com",
            "Stuard",
            "pcValidator01",
            "DustStorm",
            "icenode"
          ],
          "value": {
            "timestamp": 1652770890,
            "ledger_id": 1,
            "state_root_hash": "Dcda5BCQSGiQfCGWEhXhtm3zkum4YABTGYRrNueaWhHX",
            "txn_root_hash": "97DQXF8qNCMCwFZVmDUmHwhoY4Mm3cihbhZB9KdricFR",
            "pool_state_root_hash": "Hrp6rfmhf1jSJmnr75WTFcTnGWG2h3uYmhCyY6Da2Bp"
          }
        }
      },
      "txnTime": 1592967557,
      "reqId": 1652771098602296300,
      "data": "
{\"dest\":\"7Tqg6BwSSWapxgUDm9KKgg\",\"identifier\":\"mgtqGEF8EBQHBLREdRoJ4\",\"ro
le\":\"0\",\"seqNo\":54474,\"txnTime\":1592967557,\"verkey\":\"~AmJxMwcmQMpeugR5gh
oU8o\"}",
      "seqNo": 54474,
      "dest": "7Tqg6BwSSWapxgUDm9KKgg",
      "type": "105"
    },
    "op": "REPLY"
  },
  "attrResponse": {
    "result": {
      "identifier": "5Y9j3N2CB3Xm5GwbHaQd8m",
      "state_proof": {
        "proof_nodes":
"+Qcq+QIRoPZD2q4JbeO9UuhxrQoMu+Uylb8SdBPavfALd5Mj6KgqoA9Am1h8SLzOVmhvMfwRlpQkc/SND
9pEIXgBQbWQlRyIoHRCZuJULEpSDJ+oZJbl5x7aMA0jKP6QURq/MXdG8xN4oOlut5csynBgKasku0yOmyP
ESulJwhHYvWsmFZkwcqIjoHwbKdxMyHiW0vqVaEjgZSz3WKHACHWOUmHt3yYyee76oG40A5fOinbt1/vLR
y8F8DTboTPgZK95T5bjjcTFpSVOoExsSm7dy3iW1aTcuYV7i/lHZ72TB4BpR3O9fJC1h83eoGvguV9uLqg
DC8FuWXX1jXUuLEmy0PbkUzXhcm7zSemjoL4qJJ+GhkOax1F4Fwb5HBpUc9xKyhlvm0HVI/LGW/nIoBXhV
g33pUdFeqd3xQ8F6XH2734+dmAsnxgSq8KZP4LpoFs7AmL2tdwuHZbVRgT2D7swnR3qHt5N1afmXmnjvsS
+oHu11aOOi7VFQQ5Y5tfIloxRhmsHag6tgSicRYFj/4aZoMBmpSOmpOyUFHikpvCMs2Gkrd3vzgrYSnoXH
+4JYVpMoLrsA4drSfefDD5xuN8JVJAOPuIVNcCB8TYiyZuHUfYhoJ1knVbukHrKGX4MftrgfvIcyi/cWkG
SXmsSWXtZeeYGoABkplV/wimktKj50IXvkIjn1lNT8I8HgTqyykdJf6nvgPkBMYCAgKBxpbf5Gx3YisOPV
eoLK0p5JBHTRz9PFlnVHLCst51Q5qDjqj4wG7lWnOYtuJ/x2zgk9PmSOgF0wDjKdSMnjcokFKC5UChk7UO
QavONYMQTi2ASgNWLq+MBkfGrWmZQ4uBlpaDYQ+AxwbzMdEScK8Pml4jUWa5oyCbkM5DkK0hggOBBIqAV3
eRbKjowacy27pAPAct1g40iRYUkxz5jCc5yXnIpIaBYi2iFjcQxg4IUNc7FK9RvwI9oscivJwOIuNuHoe5
TC6ASd6JLVOfdP2kQk95sU9e6aKFqSEfAI77m6j39cxMAqqCz6f3BDHHET3ROGXrLZL3vsVED4tb3krVSI
yvgddK22qAkqrJAmKyAoznkMmLnsrA5uVrsXBIDMhJZXRDtTx4+D4CAgICA+FGAgICAgICgNgXQ2OpMJcN
7QRMPfCFw/YI4wXdl2mRJ3xrvkqFcOy6gpBSdLfdK7PY1Nkx1nSeOhcq0S4z5B8KigtD5yD9shtCAgICAg
ICAgID4saAMFqhgYm3bZ91RFi7nopbX0wzrYlCFrH1whi46pVf//YCAgKBwvI3Cclq0QnHgdt4jdRBMX/D
```

```
JOaCWaniJcXaB65ceZ4CgtXdNIm5Vg4u5TTI+03BF/LzbkI3nsbYSXboGpWY1jReAgICgTmudR958pkSF5
LCV+DI5kEew1aKq7Lupnhj2n+eamBKAgKDoPsKjg9MaBNokz7u7RzcD1pG5PVz5oLnYmTlU3OlnKoCAgPj
GuFcxZzZCd1NTV2FweGdVRG05S0tnZzoxOmI2YmY3YmM4ZDk2ZjNlYTlkMTMyYzgzYjNkYThlNzc2MGU0M
jAxMzg0ODU2NTczNzJkYjRkNmE5ODFkM2ZkOWW4a/hpuGd7ImxzbiI6NjAyNTYsImx1dCI6MTYyMTgyMzU
xOSwidmFsIjoiZDU2MDIxNzZiODgzMGJmMGE4NmI3M2QzYWZmZWE3OGZjODQ2MGNhYWFkNzBkNTI2YTYxY
mNjMDJmOTZiODZmZSJ9+QIRoL8TLgzZ00SLH9NRHFM/anNeLBqz/fmBboAoUz5WO9IPoCs6HPv6ffuO3hg
/uJ5GzXl/oGZD/NZYw+Gpj6mJ6R8BoCLhRb9fXDi1ltXY4zBRaz83mpU0y4osDpl2CwwVkExToOu7dJB1e
0VBJhxxEAWFKfmtdesOgUgTha2iL3jWSrRcoO+Ls4G8OSS2qPzkub/luvxdsyZrBMdtl0aMfvdTM6/eoBY
Vhn20J3EXFyirQzScwGYkbzZ9fd6zm5GDkqTpAh3FoBkMsXOPXmhssjnuKFTi+PoQBBZv+mshGAYEGEGH1
DCOoDBkadAXdb/VAwDIQxGSJB4jcddU609Cpy0Wbwa/+K2IoAOm3RyTLAmNpU7c7F/YnecF8zoQUJyJ/G4
BTxKvsbUYoA6VSh/YlPX/OWT47cKwAEbM17XpSg62EJQeMSl+cKvkoAdKxOl26bOTDXCMJR8GNRs1QM8qd
WPOa0Am8V5EPQmFoPnn2lCvyqtWS6DO8pTAlkHu8CHUwiTf5+q1w27vfE9NoKyAmiCNbF0SWzVcfZ6zY3u
yeZmehrohiNVhrDSNf1KooE9QHZAXk5Gp068GVXQPEssb4AjtTELJGrDuwJaLYMmEoOEL1nfXQxp3Ugqhi
D9gyS/ElFGXKR0JIw/HCOw4ENUgoN3wKRijBZNDfy2HbNCWcua4HcnXN74pHiDHl64EuMPfgA==",
        "root_hash": "Dcda5BCQSGiQfCGWEhXhtm3zkum4YABTGYRrNueaWhHX",
        "multi_signature": {
          "signature":
"RbCSjQEpzpac44mXdpn16kBQyBkFPhiSzCaykUhtQvhKtcir389UNBzJYkUMUFQAdrrXiJmampzuEpjnK
vLRiNgC6R6jtTXQuL1f6omH235wbnj79yNS2gW5VgNSXYnKE34CiMnAdBs1hkzsA7TrohZGbcXmnLAfphq
yy48Gkz5H2w",
          "participants": [
            "esatus_AG",
            "atbsovrin",
            "royal_sovrin",
            "OASFCU",
            "prosovitor",
            "VeridiumIDC",
            "danube",
            "sovrin.sicpa.com",
            "Stuard",
            "pcValidator01",
            "DustStorm",
            "icenode"
          ],
          "value": {
            "timestamp": 1652770890,
            "ledger_id": 1,
            "state_root_hash": "Dcda5BCQSGiQfCGWEhXhtm3zkum4YABTGYRrNueaWhHX",
            "txn_root_hash": "97DQXF8qNCMCwFZVmDUmHwhoY4Mm3cihbhZB9KdricFR",
            "pool_state_root_hash": "Hrp6rfmhf1jSJmnr75WTFcTnGWG2h3uYmhCyY6Da2Bp"
          }
        }
      },
      "txnTime": 1621823519,
      "reqId": 1652771098971425300,
      "data": "{\"endpoint\":{\"endpoint\":\"null\"}}",
      "raw": "endpoint",
      "seqNo": 60256,
      "dest": "7Tqg6BwSSWapxgUDm9KKgg",
      "type": "104"
    },
    "op": "REPLY"
  }
}
```

Le tutoriel indique également que sur le Sovrin Ledger, nous trouvons simplement le DID et la verkey pour générer un log DIDDoc. Cela est possible parce que la méthode Sovrin DID défini comment transformer un DID Sovrin MainNet en DIDDoc, et cela inclut un nombre d'entrées standard dérivé depuis les informations minimales stockées sur le ledger.

Exemple de résolution d'un did:web

La méthode did:web est lié à une entrée DNS en incluant le DNS dans le DID après le did:web. Prenons l'exemple suivant, "did:web:did.actor:alice", ici "did.actor" est le DNS. Le DIDDoc lié au DID se trouve à la localisation suivante "https:////did.json". Ainsi pour notre exemple, le DIDDoc se trouve ici. Essayons de résoudre le DID et regardons ce que cela donne.



Dans l'ordre, les onglets [DID DOCUMENT, RESOLUTION METADATA, DOCUMENT METADATA] comportent les informations suivantes :

```
{
  "@context": [
    "https://w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2018/v1"
  ],
```

```
    "id": "did:web:did.actor:alice",
    "publicKey": [
      {
        "id":
"did:web:did.actor:alice#z6MkrmNwty5ajKtFqc1U48oL2MMLjWjartwc5sf2AihZwXDN",
        "controller": "did:web:did.actor:alice",
        "type": "Ed25519VerificationKey2018",
        "publicKeyBase58": "DK7uJiq9PnPnj7AmNZqVBFoLuwTjT1hFPrk6LSjZ2JRz"
      }
    ],
    "authentication": [
      "did:web:did.actor:alice#z6MkrmNwty5ajKtFqc1U48oL2MMLjWjartwc5sf2AihZwXDN"
    ],
    "assertionMethod": [
      "did:web:did.actor:alice#z6MkrmNwty5ajKtFqc1U48oL2MMLjWjartwc5sf2AihZwXDN"
    ],
    "capabilityDelegation": [
      "did:web:did.actor:alice#z6MkrmNwty5ajKtFqc1U48oL2MMLjWjartwc5sf2AihZwXDN"
    ],
    "capabilityInvocation": [
      "did:web:did.actor:alice#z6MkrmNwty5ajKtFqc1U48oL2MMLjWjartwc5sf2AihZwXDN"
    ],
    "keyAgreement": [
      {
        "id":
"did:web:did.actor:alice#zC8GybikEfyNaausDA4mkT4egP7SNLx2T1d1kujLQbcP6h",
        "type": "X25519KeyAgreementKey2019",
        "controller": "did:web:did.actor:alice",
        "publicKeyBase58": "CaSHXEvLKS6SfN9aBfkVGBpp15jSnaHazqHgLHp8KZ3Y"
      }
    ]
  }
```



```
{
  "contentType": "application/did+ld+json",
  "pattern": "^(did:web:.+)$",
  "driverUrl": "http://uni-resolver-driver-did-uport:8081/1.0/identifiers/",
  "duration": 144,
  "did": {
    "didString": "did:web:did.actor:alice",
    "methodSpecificId": "did.actor:alice",
    "method": "web"
  }
}
```



```
{}
```

Le tutoriel précise également que cela fonctionne pour n'importe quelle entrée DNS, ainsi si nous avons notre propre serveur, il suffit d'ajouter un document "did.json" et on peut le résoudre en tant que DID.

Autres méthodes DID disponibles

Toutes les méthodes disponibles sur le site d'Universal Resolver et le registre des méthodes DID du W3C peuvent en théorie être utilisées sur Aries comme par exemple :

- did:btcr un DID sur le ledger Bitcoin
- did:erc725 un DID sur le ledger Ethereum
- did:ethr une autre méthode DID sur le ledger Ethereum
- did:ion un DID sur le ledger Bitcoin en utilisant une méthode DID de Microsoft
- did:ipid un DID basé sur IPFS
- did:key un DID non sur un ledger, mais sur une clé publique dans le DID

HyperLedger Aries - Lab Option de Démarrage d'un Agent

Ce document montre la reproduction du lab suivant via une machine wsl2 Ubuntu sur Windows 11.

Prérequis

Pour effectuer ce Lab, il est nécessaire d'avoir :

- Docker
- Git

Avant-Propos sur les options de démarrage d'un Agent

Un agent en Aries est un composant stateful qui persiste ses données dans son wallet et optionnellement sur un ledger. La première fois qu'un agent démarre, il n'a pas de stockage et il doit ainsi créer un wallet et tous les objets du ledger qui peut servir à ce rôle.

Quand l'agent est en développement, il est possible de le démarrer, créer son état, le stopper et le redémarrer autant de fois que nécessaire, mais lorsqu'il est en production, il est important d'initialiser son état qu'une seule fois.

Ainsi Aries offre deux modes d'opération pour lancer les agents, l'opération "provision" a pour but d'être utilisée qu'une seule fois pour établir le wallet et les ledgers si nécessaire ou alors si quelque chose doit être rajouté à l'agent, l'opération "start" assume que tout est déjà en place et démarre normalement.

Il y a diverses options de démarrage pour les agents, voici quelques exemples pour ACA-Py :

- Debug - options pour le débogage
- Admin - options pour configurer la connexion entre ACA-Py et les contrôleurs
- General - options sur les extensions qui peuvent être ajoutées et où les objets non-Indy sont stockés
- Ledger - options pour donner différentes façons pour l'agent de se connecter au ledger
- Logging - options pour le log
- Mediation - options pour configurer une instance qui est utilisée comme mediator
- Multi-tenant - options pour exécuter une instance en mode multi-tenant, cela signifie que le même agent est utilisé pour plusieurs entités différentes
- Protocol - options pour spécifier la gestion des protocoles
- Start-up - options sur les profils
- Transport - options sur les interfaces qui sont utilisées pour la connexion et le messaging avec les autres agents
- Wallet - options sur le stockage des clés, DID, objets du Indy ledger et les identifiants

La documentation nous indique que ces options, même si les noms sont ici spécifiques à ACA-Py, se trouvent également dans les autres implémentations de la Framework Aries.

Étapes de réalisation

Cloner le GitHub et afficher la version de ACA-Py

Tout d'abord, il nous faut cloner le repository de ACA-Py.

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython$ git clone
https://github.com/hyperledger/aries-cloudagent-python
...
valonrexhepi@WINDOWSDESKTOP:~/AriesPython$ cd aries-cloudagent-python/
```

Nous pouvons ensuite lancer le docker en affichant la version actuelle.

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-cloudagent-python$
./scripts/run_docker --version
...
...
0.7.4-rc2
```

Afficher l'aide sur les options de démarrage

Il est possible d'afficher l'aide pour n'importe laquelle des options de démarrage, voici l'exemple avec l'option start :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-cloudagent-python$
./scripts/run_docker start --help
...
usage: aca-py start [-h] [--admin <host> <port>] [--admin-api-key <api-key>]
                    [--admin-insecure-mode] [--no-receive-invites]
                    [--help-link <help-url>] [--webhook-url <url#api_key>]
                    [--admin-client-max-request-size
ADMIN_CLIENT_MAX_REQUEST_SIZE]
                    [--debug] [--debug-seed <debug-did-seed>]
                    [--debug-connections] [--debug-credentials]
                    [--debug-presentations] [--invite] [--connections-invite]
                    [--invite-label <label>] [--invite-multi-use]
                    [--invite-public] [--invite-metadata-json <metadata-json>]
                    [--test-suite-endpoint <endpoint>] [--auto-accept-invites]
                    [--auto-accept-requests] [--auto-respond-messages]
                    [--auto-respond-credential-proposal]
                    [--auto-respond-credential-offer]
                    [--auto-respond-credential-request]
                    [--auto-respond-presentation-proposal]
                    [--auto-respond-presentation-request]
                    [--auto-store-credential] [--auto-verify-presentation]
                    [--auto-disclose-features]
                    [--disclose-features-list DISCLOSE_FEATURES_LIST]
                    [--arg-file ARG_FILE] [--plugin <module>]
                    [--plugin-config PLUGIN_CONFIG]
                    [-o <KEY=VALUE> [<KEY=VALUE> ...]]
                    [--storage-type <storage-type>]
                    [-e <endpoint> [<endpoint> ...]]
```

```
                    [--profile-endpoint <profile_endpoint>]
                    [--read-only-ledger]
                    [--tails-server-base-url <tails-server-base-url>]
                    [--tails-server-upload-url <tails-server-upload-url>]
                    [--notify-revocation] [--monitor-revocation-notification]
                    [--ledger-pool-name <ledger-pool-name>]
                    [--genesis-transactions <genesis-transactions>]
                    [--genesis-file <genesis-file>]
                    [--genesis-url <genesis-url>] [--no-ledger]
                    [--ledger-keepalive LEDGER_KEEPALIVE]
                    [--ledger-socks-proxy <host:port>]
                    [--genesis-transactions-list <genesis-transactions-list>]
                    [--accept-taa <acceptance-mechanism> <taa-version>]
                    [--log-config <path-to-config>] [--log-file <log-file>]
                    [--log-level <log-level>] [--auto-ping-connection]
                    [--auto-accept-intro-invitation-requests]
                    [--invite-base-url <base-url>] [--monitor-ping]
                    [--monitor-forward] [--public-invites] [--timing]
                    [--timing-log <log-path>] [--trace]
                    [--trace-target <trace-target>] [--trace-tag <trace-tag>]
                    [--trace-label <trace-label>]
                    [--preserve-exchange-records] [--emit-new-didcomm-prefix]
                    [--emit-new-didcomm-mime-type]
                    [--exch-use-unencrypted-tags] [--auto-provision]
                    [-it <module> <host> <port>] [-ot <module>] [-l <label>]
                    [--image-url IMAGE_URL]
                    [--max-message-size <message-size>]
                    [--enable-undelivered-queue]
                    [--max-outbound-retry MAX_OUTBOUND_RETRY]
                    [--ws-heartbeat-interval <interval>]
                    [--ws-timeout-interval <interval>]
                    [--mediator-invitation <invite URL to mediator>]
                    [--mediator-connections-invite] [--open-mediation]
                    [--default-mediator-id <mediation id>]
                    [--clear-default-mediator] [--seed <wallet-seed>]
                    [--wallet-local-did] [--wallet-allow-insecure-seed]
                    [--wallet-key <wallet-key>]
                    [--wallet-rekey <wallet-rekey>]
                    [--wallet-name <wallet-name>]
                    [--wallet-type <wallet-type>]
                    [--wallet-storage-type <storage-type>]
                    [--wallet-storage-config <storage-config>]
                    [--wallet-key-derivation-method <key-derivation-method>]
                    [--wallet-storage-creds <storage-creds>]
                    [--replace-public-did] [--recreate-wallet] [--multitenant]
                    [--jwt-secret <jwt-secret>] [--multitenant-admin]
                    [--multitenancy-config <multitenancy-config>]
                    [--endorser-protocol-role <endorser-role>]
                    [--endorser-invitation <endorser-invitation>]
                    [--endorser-public-did <endorser-public-did>]
                    [--endorser-endorse-with-did <endorser-endorse-with-did>]
                    [--endorser-alias <endorser-alias>]
                    [--auto-request-endorsement] [--auto-endorse-transactions]
                    [--auto-write-transactions]
```

```
                    [--auto-create-revocation-transactions]
                    [--auto-promote-author-did]

optional arguments:
  -h, --help            show this help message and exit

Admin:
  --admin <host> <port>
                        Specify the host and port on which to run the
                        administrative server. If not provided, no admin
                        server is made available. [env var: ACAPY_ADMIN]
  --admin-api-key <api-key>
                        Protect all admin endpoints with the provided API key.
                        API clients (e.g. the controller) must pass the key in
                        the HTTP header using 'X-API-Key: <api key>'. Either
                        this parameter or the '--admin-insecure-mode'
                        parameter MUST be specified. [env var:
                        ACAPY_ADMIN_API_KEY]
  --admin-insecure-mode
                        Run the admin web server in insecure mode. DO NOT USE
                        FOR PRODUCTION DEPLOYMENTS. The admin server will be
                        publicly available to anyone who has access to the
                        interface. Either this parameter or the '--api-key'
                        parameter MUST be specified. [env var:
                        ACAPY_ADMIN_INSECURE_MODE]
  --no-receive-invites  Prevents an agent from receiving invites by removing
                        the '/connections/receive-invite' route from the
                        administrative interface. Default: false. [env var:
                        ACAPY_NO_RECEIVE_INVITES]
  --help-link <help-url>
                        A URL to an administrative interface help web page
                        that a controller user interface can get from the
                        agent and provide as a link to users. [env var:
                        ACAPY_HELP_LINK]
  --webhook-url <url#api_key>
                        Send webhooks containing internal state changes to the
                        specified URL. Optional API key to be passed in the
                        request body can be appended using a hash separator
                        [#]. This is useful for a controller to monitor agent
                        events and respond to those events using the admin
                        API. If not specified, webhooks are not published by
                        the agent. [env var: ACAPY_WEBHOOK_URL]
  --admin-client-max-request-size ADMIN_CLIENT_MAX_REQUEST_SIZE
                        Maximum client request size to admin server, in
                        megabytes: default 1 [env var:
                        ACAPY_ADMIN_CLIENT_MAX_REQUEST_SIZE]

Debug:
  --debug               Enables a remote debugging service that can be
                        accessed using ptvsd for Visual Studio Code. The
                        Framework will wait for the debugger to connect at
                        start-up. Default: false. [env var: ACAPY_DEBUG]
  --debug-seed <debug-did-seed>
                        Specify the debug seed to use. [env var:
```

```
                        ACAPY_DEBUG_SEED]
    --debug-connections   Enable additional logging around connections. Default:
                        false. [env var: ACAPY_DEBUG_CONNECTIONS]
    --debug-credentials   Enable additional logging around credential exchanges.
                        Default: false. [env var: ACAPY_DEBUG_CREDENTIALS]
    --debug-presentations
                        Enable additional logging around presentation
                        exchanges. Default: false. [env var:
                        ACAPY_DEBUG_PRESENTATIONS]
    --invite              After startup, generate and print a new out-of-band
                        connection invitation URL. Default: false. [env var:
                        ACAPY_INVITE]
    --connections-invite  After startup, generate and print a new connections
                        protocol style invitation URL. Default: false. [env
                        var: ACAPY_CONNECTIONS_INVITE]
    --invite-label <label>
                        Specify the label of the generated invitation. [env
                        var: ACAPY_INVITE_LABEL]
    --invite-multi-use    Flag specifying the generated invite should be multi-
                        use. [env var: ACAPY_INVITE_MULTI_USE]
    --invite-public       Flag specifying the generated invite should be public.
                        [env var: ACAPY_INVITE_PUBLIC]
    --invite-metadata-json <metadata-json>
                        Add metadata json to invitation created with --invite
                        argument. [env var: ACAPY_INVITE_METADATA_JSON]
    --test-suite-endpoint <endpoint>
                        URL endpoint for sending messages to the test suite
                        agent. [env var: ACAPY_TEST_SUITE_ENDPOINT]
    --auto-accept-invites
                        Automatically accept invites without firing a webhook
                        event or waiting for an admin request. Default: false.
                        [env var: ACAPY_AUTO_ACCEPT_INVITES]
    --auto-accept-requests
                        Automatically accept connection requests without
                        firing a webhook event or waiting for an admin
                        request. Default: false. [env var:
                        ACAPY_AUTO_ACCEPT_REQUESTS]
    --auto-respond-messages
                        Automatically respond to basic messages indicating the
                        message was received. Default: false. [env var:
                        ACAPY_AUTO_RESPOND_MESSAGES]
    --auto-respond-credential-proposal
                        Auto-respond to credential proposals with
                        corresponding credential offers [env var:
                        ACAPY_AUTO_RESPOND_CREDENTIAL_PROPOSAL]
    --auto-respond-credential-offer
                        Automatically respond to Indy credential offers with a
                        credential request. Default: false [env var:
                        ACAPY_AUTO_RESPOND_CREDENTIAL_OFFER]
    --auto-respond-credential-request
                        Auto-respond to credential requests with corresponding
                        credentials [env var:
                        ACAPY_AUTO_RESPOND_CREDENTIAL_REQUEST]
    --auto-respond-presentation-proposal
```

```
                        Auto-respond to presentation proposals with
                        corresponding presentation requests [env var:
                        ACAPY_AUTO_RESPOND_PRESENTATION_PROPOSAL]
  --auto-respond-presentation-request
                        Automatically respond to Indy presentation requests
                        with a constructed presentation if a corresponding
                        credential can be retrieved for every referent in the
                        presentation request. Default: false. [env var:
                        ACAPY_AUTO_RESPOND_PRESENTATION_REQUEST]
  --auto-store-credential
                        Automatically store an issued credential upon receipt.
                        Default: false. [env var: ACAPY_AUTO_STORE_CREDENTIAL]
  --auto-verify-presentation
                        Automatically verify a presentation when it is
                        received. Default: false. [env var:
                        ACAPY_AUTO_VERIFY_PRESENTATION]

Discover features:
  --auto-disclose-features
                        Specifies that the agent will proactively/auto
                        disclose protocols and goal-codes features on
                        connection creation [RFC0557]. [env var:
                        ACAPY_AUTO_DISCLOSE_FEATURES]
  --disclose-features-list DISCLOSE_FEATURES_LIST
                        Load YAML file path that specifies which features to
                        disclose. [env var: ACAPY_DISCLOSE_FEATURES_LIST]

General:
  --arg-file ARG_FILE   Load aca-py arguments from the specified file. Note
                        that this file *must* be in YAML format.
  --plugin <module>     Load <module> as external plugin module. Multiple
                        instances of this parameter can be specified. [env
                        var: ACAPY_PLUGIN]
  --plugin-config PLUGIN_CONFIG
                        Load YAML file path that defines external plugin
                        configuration. [env var: ACAPY_PLUGIN_CONFIG]
  -o <KEY=VALUE> [<KEY=VALUE> ...], --plugin-config-value <KEY=VALUE> [<KEY=VALUE>
...]
                        Set an arbitrary plugin configuration option in the
                        format KEY=VALUE. Use dots in KEY to set deeply nested
                        values, as in "a.b.c=value". VALUE is parsed as yaml.
  --storage-type <storage-type>
                        Specifies the type of storage provider to use for the
                        internal storage engine. This storage interface is
                        used to store internal state Supported internal
                        storage types are 'basic' (memory) and 'indy'. The
                        default (if not specified) is 'indy' if the wallet
                        type is set to 'indy', otherwise 'basic'. [env var:
                        ACAPY_STORAGE_TYPE]
  -e <endpoint> [<endpoint> ...], --endpoint <endpoint> [<endpoint> ...]
                        Specifies the endpoints to put into DIDDocs to inform
                        other agents of where they should send messages
                        destined for this agent. Each endpoint could be one of
                        the specified inbound transports for this agent, or
```

```
                        the endpoint could be that of another agent (e.g.
                        'https://example.com/agent-endpoint') if the routing
                        of messages to this agent by a mediator is configured.
                        The first endpoint specified will be used in
                        invitations. The endpoints are used in the formation
                        of a connection with another agent. [env var:
                        ACAPY_ENDPOINT]
  --profile-endpoint <profile_endpoint>
                        Specifies the profile endpoint for the (public) DID.
                        [env var: ACAPY_PROFILE_ENDPOINT]
  --read-only-ledger    Sets ledger to read-only to prevent updates. Default:
                        false. [env var: ACAPY_READ_ONLY_LEDGER]

Revocation:
  --tails-server-base-url <tails-server-base-url>
                        Sets the base url of the tails server in use. [env
                        var: ACAPY_TAILS_SERVER_BASE_URL]
  --tails-server-upload-url <tails-server-upload-url>
                        Sets the base url of the tails server for upload,
                        defaulting to the tails server base url. [env var:
                        ACAPY_TAILS_SERVER_UPLOAD_URL]
  --notify-revocation   Specifies that aca-py will notify credential
                        recipients when revoking a credential it issued. [env
                        var: ACAPY_NOTIFY_REVOCATION]
  --monitor-revocation-notification
                        Specifies that aca-py will emit webhooks on
                        notification of revocation received. [env var:
                        ACAPY_MONITOR_REVOCATION_NOTIFICATION]

Ledger:
  --ledger-pool-name <ledger-pool-name>
                        Specifies the name of the indy pool to be opened. This
                        is useful if you have multiple pool configurations.
                        [env var: ACAPY_LEDGER_POOL_NAME]
  --genesis-transactions <genesis-transactions>
                        Specifies the genesis transactions to use to connect
                        to a Hyperledger Indy ledger. The transactions are
                        provided as string of JSON e.g.
                        '{"reqSignature":{},"txn":{"data":{"d... <snip>}}}'
                        [env var: ACAPY_GENESIS_TRANSACTIONS]
  --genesis-file <genesis-file>
                        Specifies a local file from which to read the genesis
                        transactions. [env var: ACAPY_GENESIS_FILE]
  --genesis-url <genesis-url>
                        Specifies the url from which to download the genesis
                        transactions. For example, if you are using 'von-
                        network', the URL might be
                        'http://localhost:9000/genesis'. Genesis transactions
                        URLs are available for the Sovrin test/main networks.
                        [env var: ACAPY_GENESIS_URL]
  --no-ledger           Specifies that aca-py will run with no ledger
                        configured. This must be set if running in no-ledger
                        mode. Overrides any specified ledger or genesis
                        configurations. Default: false. [env var:
```

```
                        ACAPY_NO_LEDGER]
  --ledger-keepalive LEDGER_KEEPALIVE
                        Specifies how many seconds to keep the ledger open.
                        Default: 5 [env var: ACAPY_LEDGER_KEEP_ALIVE]
  --ledger-socks-proxy <host:port>
                        Specifies the socks proxy (NOT http proxy) hostname
                        and port in format 'hostname:port'. This is an
                        optional parameter to be passed to ledger pool
                        configuration and ZMQ in case if aca-py is running in
                        a corporate/private network behind a corporate proxy
                        and will connect to the public (outside of corporate
                        network) ledger pool [env var:
                        ACAPY_LEDGER_SOCKS_PROXY]
  --genesis-transactions-list <genesis-transactions-list>
                        Load YAML configuration for connecting to multiple
                        HyperLedger Indy ledgers. [env var:
                        ACAPY_GENESIS_TRANSACTIONS_LIST]
  --accept-taa <acceptance-mechanism> <taa-version>
                        Specify the acceptance mechanism and taa version for
                        which to accept the transaction author agreement. If
                        not provided, the TAA must be accepted through the TTY
                        or the admin API. [env var: ACAPY_ACCEPT_TAA]

Logging:
  --log-config <path-to-config>
                        Specifies a custom logging configuration file [env
                        var: ACAPY_LOG_CONFIG]
  --log-file <log-file>
                        Overrides the output destination for the root logger
                        (as defined by the log config file) to the named <log-
                        file>. [env var: ACAPY_LOG_FILE]
  --log-level <log-level>
                        Specifies a custom logging level as one of: ('debug',
                        'info', 'warning', 'error', 'critical') [env var:
                        ACAPY_LOG_LEVEL]

Protocol:
  --auto-ping-connection
                        Automatically send a trust ping immediately after a
                        connection response is accepted. Some agents require
                        this before marking a connection as 'active'. Default:
                        false. [env var: ACAPY_AUTO_PING_CONNECTION]
  --auto-accept-intro-invitation-requests
                        Automatically accept introduction invitations.
                        Default: false. [env var:
                        ACAPY_AUTO_ACCEPT_INTRO_INVITATION_REQUESTS]
  --invite-base-url <base-url>
                        Base URL to use when formatting connection invitations
                        in URL format. [env var: ACAPY_INVITE_BASE_URL]
  --monitor-ping        Send a webhook when a ping is sent or received. [env
                        var: ACAPY_MONITOR_PING]
  --monitor-forward     Send a webhook when a forward is received. [env var:
                        ACAPY_MONITOR_FORWARD]
  --public-invites      Send invitations out, and receive connection requests,
```

```
                        using the public DID for the agent. Default: false.
                        [env var: ACAPY_PUBLIC_INVITES]
  --timing              Include timing information in response messages. [env
                        var: ACAPY_TIMING]
  --timing-log <log-path>
                        Write timing information to a given log file. [env
                        var: ACAPY_TIMING_LOG]
  --trace               Generate tracing events. [env var: ACAPY_TRACE]
  --trace-target <trace-target>
                        Target for trace events ("log", "message", or http
                        endpoint). [env var: ACAPY_TRACE_TARGET]
  --trace-tag <trace-tag>
                        Tag to be included when logging events. [env var:
                        ACAPY_TRACE_TAG]
  --trace-label <trace-label>
                        Label (agent name) used logging events. [env var:
                        ACAPY_TRACE_LABEL]
  --preserve-exchange-records
                        Keep credential exchange records after exchange has
                        completed. [env var: ACAPY_PRESERVE_EXCHANGE_RECORDS]
  --emit-new-didcomm-prefix
                        Emit protocol messages with new DIDComm prefix; i.e.,
                        'https://didcomm.org/' instead of (default) prefix
                        'did:sov:BzCbsNYhMrjHiqZDTUASHg;spec/'. [env var:
                        ACAPY_EMIT_NEW_DIDCOMM_PREFIX]
  --emit-new-didcomm-mime-type
                        Send packed agent messages with the DIDComm MIME type
                        as of RFC 0044; i.e., 'application/didcomm-envelope-
                        enc' instead of 'application/ssi-agent-wire'. [env
                        var: ACAPY_EMIT_NEW_DIDCOMM_MIME_TYPE]
  --exch-use-unencrypted-tags
                        Store tags for exchange protocols (credential and
                        presentation) using unencrypted rather than encrypted
                        tags [env var: ACAPY_EXCH_USE_UNENCRYPTED_TAGS]

Start-up:
  --auto-provision      If the requested profile does not exist, initialize it
                        with the given parameters. [env var:
                        ACAPY_AUTO_PROVISION]

Transport:
  -it <module> <host> <port>, --inbound-transport <module> <host> <port>
                        REQUIRED. Defines the inbound transport(s) on which
                        the agent listens for receiving messages from other
                        agents. This parameter can be specified multiple times
                        to create multiple interfaces. Built-in inbound
                        transport types include 'http' and 'ws'. However,
                        other transports can be loaded by specifying an
                        absolute module path. [env var:
                        ACAPY_INBOUND_TRANSPORT]
  -ot <module>, --outbound-transport <module>
                        REQUIRED. Defines the outbound transport(s) on which
                        the agent will send outgoing messages to other agents.
                        This parameter can be passed multiple times to supoort
```

```
                        multiple transport types. Supported outbound transport
                        types are 'http' and 'ws'. [env var:
                        ACAPY_OUTBOUND_TRANSPORT]
  -l <label>, --label <label>
                        Specifies the label for this agent. This label is
                        publicized (self-attested) to other agents as part of
                        forming a connection. [env var: ACAPY_LABEL]
  --image-url IMAGE_URL
                        Specifies the image url for this agent. This image url
                        is publicized (self-attested) to other agents as part
                        of forming a connection. [env var: ACAPY_IMAGE_URL]
  --max-message-size <message-size>
                        Set the maximum size in bytes for inbound agent
                        messages. [env var: ACAPY_MAX_MESSAGE_SIZE]
  --enable-undelivered-queue
                        Enable the outbound undelivered queue that enables
                        this agent to hold messages for delivery to agents
                        without an endpoint. This option will require
                        additional memory to store messages in the queue. [env
                        var: ACAPY_ENABLE_UNDELIVERED_QUEUE]
  --max-outbound-retry MAX_OUTBOUND_RETRY
                        Set the maximum retry number for undelivered outbound
                        messages. Increasing this number might cause to
                        increase the accumulated messages in message queue.
                        Default value is 4. [env var:
                        ACAPY_MAX_OUTBOUND_RETRY]
  --ws-heartbeat-interval <interval>
                        When using Websocket Inbound Transport, send WS pings
                        every <interval> seconds. [env var:
                        ACAPY_WS_HEARTBEAT_INTERVAL]
  --ws-timeout-interval <interval>
                        When using Websocket Inbound Transport, timeout the WS
                        connection after <interval> seconds without a
                        heartbeat ping. [env var: ACAPY_WS_TIMEOUT_INTERVAL]

Mediation invitation:
  --mediator-invitation <invite URL to mediator>
                        Connect to mediator through provided invitation and
                        send mediation request and set as default mediator.
                        [env var: ACAPY_MEDIATION_INVITATION]
  --mediator-connections-invite
                        Connect to mediator through a connection invitation.
                        If not specified, connect using an OOB invitation.
                        Default: false. [env var:
                        ACAPY_MEDIATION_CONNECTIONS_INVITE]

Mediation:
  --open-mediation      Enables automatic granting of mediation. After
                        establishing a connection, if enabled, an agent may
                        request message mediation and be granted it
                        automatically, which will allow the mediator to
                        forward messages on behalf of the recipient. See
                        aries-rfc:0211. [env var: ACAPY_MEDIATION_OPEN]
  --default-mediator-id <mediation id>
```

```
                        Set the default mediator by ID [env var:
                        ACAPY_DEFAULT_MEDIATION_ID]
  --clear-default-mediator
                        Clear the stored default mediator. [env var:
                        ACAPY_CLEAR_DEFAULT_MEDIATOR]

Wallet:
  --seed <wallet-seed>  Specifies the seed to use for the creation of a public
                        DID for the agent to use with a Hyperledger Indy
                        ledger, or a local ('--wallet-local-did') DID. If
                        public, the DID must already exist on the ledger. [env
                        var: ACAPY_WALLET_SEED]
  --wallet-local-did    If this parameter is set, provisions the wallet with a
                        local DID from the '--seed' parameter, instead of a
                        public DID to use with a Hyperledger Indy ledger. [env
                        var: ACAPY_WALLET_LOCAL_DID]
  --wallet-allow-insecure-seed
                        If this parameter is set, allows to use a custom seed
                        to create a local DID [env var:
                        ACAPY_WALLET_ALLOW_INSECURE_SEED]
  --wallet-key <wallet-key>
                        Specifies the master key value to use to open the
                        wallet. [env var: ACAPY_WALLET_KEY]
  --wallet-rekey <wallet-rekey>
                        Specifies a new master key value to which to rotate
                        and to open the wallet next time. [env var:
                        ACAPY_WALLET_REKEY]
  --wallet-name <wallet-name>
                        Specifies the wallet name to be used by the agent.
                        This is useful if your deployment has multiple
                        wallets. [env var: ACAPY_WALLET_NAME]
  --wallet-type <wallet-type>
                        Specifies the type of Indy wallet provider to use.
                        Supported internal storage types are 'basic' (memory)
                        and 'indy'. The default (if not specified) is 'basic'.
                        [env var: ACAPY_WALLET_TYPE]
  --wallet-storage-type <storage-type>
                        Specifies the type of Indy wallet backend to use.
                        Supported internal storage types are 'basic' (memory),
                        'default' (sqlite), and 'postgres_storage'. The
                        default, if not specified, is 'default'. [env var:
                        ACAPY_WALLET_STORAGE_TYPE]
  --wallet-storage-config <storage-config>
                        Specifies the storage configuration to use for the
                        wallet. This is required if you are for using
                        'postgres_storage' wallet storage type. For example,
                        '{"url":"localhost:5432",
                        "wallet_scheme":"MultiWalletSingleTable"}'. This
                        configuration maps to the indy sdk postgres plugin
                        (PostgresConfig). [env var:
                        ACAPY_WALLET_STORAGE_CONFIG]
  --wallet-key-derivation-method <key-derivation-method>
                        Specifies the key derivation method used for wallet
                        encryption.If RAW key derivation method is used, also
```

```
                        --wallet-key parameter is expected. [env var:
                        ACAPY_WALLET_KEY_DERIVATION_METHOD]
  --wallet-storage-creds <storage-creds>
                        Specifies the storage credentials to use for the
                        wallet. This is required if you are for using
                        'postgres_storage' wallet For example,
                        '{"account":"postgres","password":
                        "mysecretpassword","admin_account":"postgres",
                        "admin_password":"mysecretpassword"}'. This
                        configuration maps to the indy sdk postgres plugin
                        (PostgresCredentials). NOTE: admin_user must have the
                        CREATEDB role or else initialization will fail. [env
                        var: ACAPY_WALLET_STORAGE_CREDS]
  --replace-public-did  If this parameter is set and an agent already has a
                        public DID, and the '--seed' parameter specifies a new
                        DID, the agent will use the new DID in place of the
                        existing DID. Default: false. [env var:
                        ACAPY_REPLACE_PUBLIC_DID]
  --recreate-wallet     If an existing wallet exists with the same name,
                        remove and recreate it during provisioning. [env var:
                        ACAPY_RECREATE_WALLET]

Multitenant:
  --multitenant         Enable multitenant mode. [env var: ACAPY_MULTITENANT]
  --jwt-secret <jwt-secret>
                        Specify the secret to be used for Json Web Token (JWT)
                        creation and verification. The JWTs are used to
                        authenticate and authorize multitenant wallets. [env
                        var: ACAPY_MULTITENANT_JWT_SECRET]
  --multitenant-admin   Specify whether to enable the multitenant admin api.
                        [env var: ACAPY_MULTITENANT_ADMIN]
  --multitenancy-config <multitenancy-config>
                        Specify multitenancy configuration ("wallet_type" and
                        "wallet_name"). For example: "{"wallet_type":"askar-
                        profile","wallet_name":"askar-profile-name",
                        "key_derivation_method":"RAW"}""wallet_name" is only
                        used when "wallet_type" is "askar-profile" [env var:
                        ACAPY_MULTITENANCY_CONFIGURATION]

Endorsement:
  --endorser-protocol-role <endorser-role>
                        Specify the role ('author' or 'endorser') which this
                        agent will participate. Authors will request
                        transaction endorement from an Endorser. Endorsers
                        will endorse transactions from Authors, and may write
                        their own transactions to the ledger. If no role (or
                        'none') is specified then the endorsement protocol
                        will not be used and this agent will write
                        transactions to the ledger directly. [env var:
                        ACAPY_ENDORSER_ROLE]
  --endorser-invitation <endorser-invitation>
                        For transaction Authors, specify the invitation used
                        to connect to the Endorser agent who will be endorsing
                        transactions. Note this is a multi-use invitation
```

```
                        created by the Endorser agent. [env var:
                        ACAPY_ENDORSER_INVITATION]
  --endorser-public-did <endorser-public-did>
                        For transaction Authors, specify the public DID of the
                        Endorser agent who will be endorsing transactions.
                        [env var: ACAPY_ENDORSER_PUBLIC_DID]
  --endorser-endorse-with-did <endorser-endorse-with-did>
                        For transaction Endorsers, specify the DID to use to
                        endorse transactions. The default (if not specified)
                        is to use the Endorser's Public DID. [env var:
                        ACAPY_ENDORSER_ENDORSE_WITH_DID]
  --endorser-alias <endorser-alias>
                        For transaction Authors, specify the alias of the
                        Endorser connection that will be used to endorse
                        transactions. [env var: ACAPY_ENDORSER_ALIAS]
  --auto-request-endorsement
                        For Authors, specify whether to automatically request
                        endorsement for all transactions. (If not specified,
                        the controller must invoke the request endorse
                        operation for each transaction.) [env var:
                        ACAPY_AUTO_REQUEST_ENDORSEMENT]
  --auto-endorse-transactions
                        For Endorsers, specify whether to automatically
                        endorse any received endorsement requests. (If not
                        specified, the controller must invoke the endorsement
                        operation for each transaction.) [env var:
                        ACAPY_AUTO_ENDORSE_TRANSACTIONS]
  --auto-write-transactions
                        For Authors, specify whether to automatically write
                        any endorsed transactions. (If not specified, the
                        controller must invoke the write transaction operation
                        for each transaction.) [env var:
                        ACAPY_AUTO_WRITE_TRANSACTIONS]
  --auto-create-revocation-transactions
                        For Authors, specify whether to automatically create
                        transactions for a cred def's revocation registry. (If
                        not specified, the controller must invoke the
                        endpoints required to create the revocation registry
                        and assign to the cred def.) [env var:
                        ACAPY_CREATE_REVOCATION_TRANSACTIONS]
  --auto-promote-author-did
                        For Authors, specify whether to automatically promote
                        a DID to the wallet public DID after writing to the
                        ledger. [env var: ACAPY_PROMOTE-AUTHOR-DID]

Args that start with '--' (eg. --admin) can also be set in a config file
(specified via --arg-file). The config file uses YAML syntax and must
represent a YAML 'mapping' (for details, see
http://learn.getgrav.org/advanced/yaml). If an arg is specified in more than
one place, then commandline values override environment variables which
override config file values which override defaults
```

Différentes méthodes

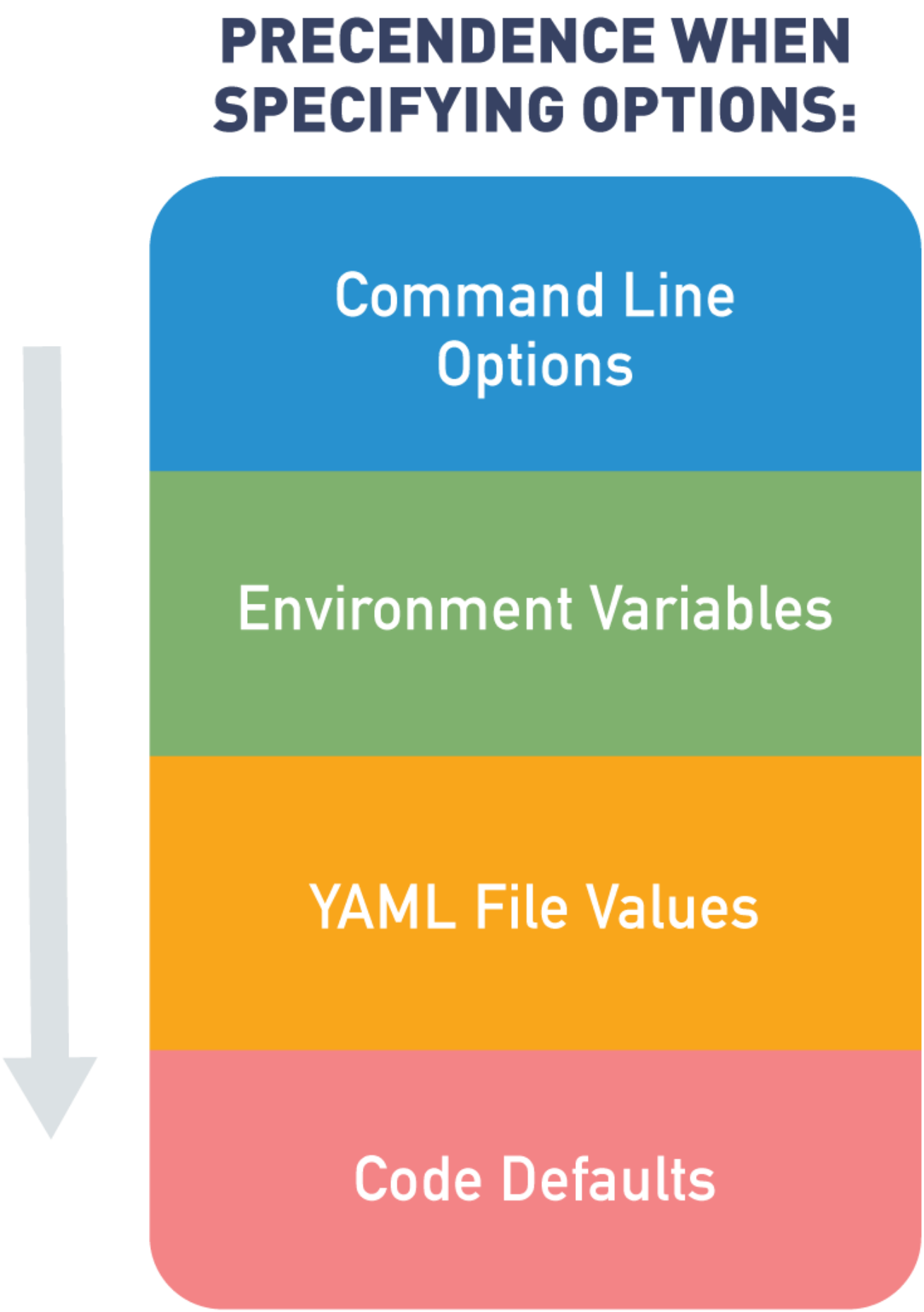
Comme nous le précise la documentation, il y a plusieurs façons de passer les options de démarrage.La première est celle via la ligne de commande, comme nous venons de le voir.

La deuxième est de faire des variables d'environnements par exemple, l'option "--genesis-file" peut être définie via une variable d'environnement nommée "ACAPY_GENESIS_FILE".

La troisième est en utilisant un fichier de configuration YAML, voici par exemple un exemple du contenu de ce fichier :

```
# Use the Indy BCovrin Test genesis file:
genesis-file: http://test.bcovrin.vonx.io/genesis
```

Attention cependant, si nous indiquons des options de différentes façons, il y a un ordre de précédence qui est appliqué. Voici une image qui résume cet ordre :


PRECENDENCE WHEN SPECIFYING OPTIONS:
Command Line Options
Environment Variables
YAML File Values
Code Defaults

Hyperledger Aries - Lab Option de Démarrage d'un Agent

Ce document montre la reproduction du lab suivant sur une machine Windows 11.

Prérequis

Pour effectuer ce Lab, il est nécessaire d'avoir :

- Docker

Avant-Propos

Attention, l'exécution de ce tutoriel peut être effectué de plusieurs façons différentes, il y a de plus amples informations sur ce lien. Cependant pour la version "Running in Docker", il semble que l'intégration Docker avec wsl2 demande des paramétrages supplémentaires et ne fonctionne pas. Alors pour ce tutoriel, l'exécution sera faite depuis directement un terminal bash de ma machine Windows 11 hôte.

Étapes de réalisation

Nous partons du principe que le github suivant a déjà été cloner sur la machine.

Démarrer une instance de VON Network

Pour les premières étapes de VON Network, nous partons du principe que le Lab1 - LabUsingVonNetwork a été suivi. Nous allons donc "build" et "start" une instance de VON Network :

```
Valon@WINDOWSDESKTOP MINGW64 ~/Desktop/Test/AriesPython/von-network (main)
$ ./manage build
...
Valon@WINDOWSDESKTOP MINGW64 ~/Desktop/Test/AriesPython/von-network (main)
$ ./manage start --logs
...
```

Une fois que VON Network est lancé, nous devons nous rendre dans le dossier "demo" du repository d'ACA-Py pour faire la suite.

Lancer l'Agent Faber

Une fois dans le dossier "demo", nous lançons l'agent Faber comme ceci :

```
Valon@WINDOWSDESKTOP MINGW64 ~/Desktop/Test/AriesPython/aries-cloudagent-
python/demo (main)
$ ./run_demo faber
...
#1 Provision an agent and wallet, get back configuration details
```

```
Started webhook listener on port: 8022
Faber      | Registering faber.agent ...
Faber      | nym_info: {'did': 'VCzTt2b3Ts33ogFT3vsejR', 'seed':
'd_0000000000000000000000000972071', 'verkey': 'GNftnCkFV2d
wrghbdypFZpHX2qPZy2Nh7SdTib2XY7EG'}
Faber      | Registered DID: VCzTt2b3Ts33ogFT3vsejR
Created public DID
Faber      | ['/home/indy/.pyenv/versions/3.6.13/bin/python', '-m',
'aries_cloudagent', 'start', '--endpoint', 'http://host
.docker.internal:8020', '--label', 'faber.agent', '--auto-ping-connection', '--
auto-respond-messages', '--inbound-transport
', 'http', '0.0.0.0', '8020', '--outbound-transport', 'http', '--admin',
'0.0.0.0', '8021', '--admin-insecure-mode', '--wal
let-type', 'indy', '--wallet-name', 'faber.agent972071', '--wallet-key',
'faber.agent972071', '--preserve-exchange-records'
, '--auto-provision', '--public-invites', '--emit-new-didcomm-prefix', '--genesis-
transactions', '{"reqSignature":{},"txn":
{"data":{"data":
{"alias":"Node1","blskey":"4N8aUNHSgjQVgkpm8nhNEfDf6txHznoYREg9kirmJrkivgL4oSEimFF
6nsQ6M41QvhM2Z33nves5vfSn
9n1UwNFJBYtWVnHYMATn76vLuL3zU88KyeAYcHfsih3He6UHcXDxcaecHVz6jhCYz1P2UZn2bDVruL5wXp
ehgBfBaLKm3Ba","blskey_pop":"RahHYiCvoNCt
PTrVtP7nMC5eTYrsUA8WjXbdhNc8debh1agE9bGiJxWBXYNFbnJXoXhWFMvyqhqhRoq737YQemH5ik9oL7
R4NTTCz2LEZhkgLJzB3QRQqJyBNyv7acbdHrAT8nQ
9UkLbaVL9NBpnWXBTw4LEMePaSHEw66RzPNdAX1","client_ip":"host.docker.internal","clien
t_port":9702,"node_ip":"host.docker.inter
nal","node_port":9701,"services":
["VALIDATOR"]},"dest":"Gw6pDLhcBcoQesN72qfotTgFa7cbuqZpkX3Xo6pLhPhv"},"metadata":
{"from":"
Th7MpTaRZVRYnPiabds81Y"},"type":"0"},"txnMetadata":
{"seqNo":1,"txnId":"fea82e10e894419fe2bea7d96296a6d46f50f93f9eeda954ec46
1b2ed2950b62"},"ver":"1"}\n{"reqSignature":{},"txn":{"data":{"data":
{"alias":"Node2","blskey":"37rAPpXVoxzKhz7d9gkUe52XuXry
uLXoM6P6LbWDB7LSbG62Lsb33sfG7zqS8TK1MXwuCHj1FKNzVpsnafmqLG1vXN88rt38mNFs9TENzm4QHd
BzsvCuoBnPH7rpYYDo9DZNJePaDvRvqJKByCabubJ
z3XXKbEeshzpz4Ma5QYpJqjk","blskey_pop":"Qr658mWZ2YC8JXGXwMDQTzuZCWF7NK9EwxphGmcBvC
h6ybUuLxbG65nsX4JvD4SPNtkJ2w9ug1yLTj6fgmu
Dg41TgECXjLCij3RMsV8CwewBVgVN67wsA45DFWvqvLtu4rjNnE9JbdFTc1Z4WCPA3Xan44K1HoHAq9EVe
aRYs8zoF5","client_ip":"host.docker.inter
nal","client_port":9704,"node_ip":"host.docker.internal","node_port":9703,"service
s":["VALIDATOR"]},"dest":"8ECVSk179mjsjKR
LWiQtssMLgp6EPhWXtaYyStWPSGAb"},"metadata":
{"from":"EbP4aYNeTHL6q385GuVpRV"},"type":"0"},"txnMetadata":{"seqNo":2,"txnId":"
1ac8aece2a18ced660fef8694b61aac3af08ba875ce3026a160acbc3a3af35fc"},"ver":"1"}\n{"r
eqSignature":{},"txn":{"data":{"data":{"a
lias":"Node3","blskey":"3WFpdbg7C5cnLYZwFZevJqhubkFALBfCBBok15GdrKMUhUjGsk3jV6QKj6
MZgEubF7oqCafxNdkm7eswgA4sdKTRc82tLGzZBd6
vNqU8dupzup6uYUf32KTHTPQbuUM8Yk4QFXjEf2Usu2TJcNkdgpyeUSX42u5LqdDDpNSWUK5deC5","bls
key_pop":"QwDeb2CkNSx6r8QC8vGQK3GRv7Yndn8
4TGNijX8YXHPiagXajyfTjoR87rXUu4G4QLk2cF8NNyqWiYMus1623dELWwx57rLCFqGh7N4ZRbGDRP4fn
VcaKg1BcUxQ866Ven4gw8y4N56S5HzxXNBZtLYmhG
HvDtk6PFkFwCvxYrNYjh","client_ip":"host.docker.internal","client_port":9706,"node_
ip":"host.docker.internal","node_port":97
05,"services":
```

```
["VALIDATOR"]},"dest":"DKVxG2fXXTU8yT5N7hGEbXB3dfdAnYv1JczDUHpmDxya"},"metadata":
{"from":"4cU41vWW82ArfxJxHkz
XPG"},"type":"0"},"txnMetadata":
{"seqNo":3,"txnId":"7e9f355dffa78ed24668f0e0e369fd8c224076571c51e2ea8be5f26479edeb
e4"},"ver
":"1"}\n{"reqSignature":{},"txn":{"data":{"data":
{"alias":"Node4","blskey":"2zN3bHM1m4rLz54MJHYSwvqzPchYp8jkHswveCLAEJVcX6M
m1wHQD1SkPYMzUDTZvWvhuE6VNAkK3KxVeEmsanSmvjVkReDeBEMxeDaayjcZjFGPydyey1qxBHmTvAnBK
oPydvuTAqx5f7YNNRAdeLmUi99gERUU7TD8KfAa6M
pQ9bw","blskey_pop":"RPLagxaR5xdimFzwmzYnz4ZhWtYQEj8iR5ZU53T2gitPCyCHQneUn2Huc4oeL
d2B2HzkGnjAff4hWTJT6C7qHYB1Mv2wU5iHHGFWkh
nTX9WsEAbunJCV2qcaXScKj4tTfvdDKfLiVuU2av6hbsMztirRze7LvYBkRHV3tGwyCptsrP","client_
ip":"host.docker.internal","client_port":
9708,"node_ip":"host.docker.internal","node_port":9707,"services":
["VALIDATOR"]},"dest":"4PS3EDQ3dW1tci1Bp6543CfuuebjFrg36k
LAUcskGfaA"},"metadata":
{"from":"TWwCRQRZ2ZHMJFn9TzLp7W"},"type":"0"},"txnMetadata":
{"seqNo":4,"txnId":"aa5e817d7cc626170ec
a175822029339a444eb0ee8f0bd20d3b0b76e566fb008"},"ver":"1"}\n', '--seed',
'd_000000000000000000000000972071', '--webhook-url
', 'http://host.docker.internal:8022/webhooks', '--monitor-revocation-
notification', '--trace-target', 'log', '--trace-tag'
, 'acapy.events', '--trace-label', 'faber.agent.trace', '--auto-accept-invites',
'--auto-accept-requests', '--auto-store-cr
edential']
Faber      |
Faber      | ::::::::::::::::::::::::::::::::::::::::::::
Faber      | :: faber.agent                           ::
Faber      | ::                                       ::
Faber      | ::                                       ::
Faber      | :: Inbound Transports:                   ::
Faber      | ::                                       ::
Faber      | ::   - http://0.0.0.0:8020               ::
Faber      | ::                                       ::
Faber      | :: Outbound Transports:                  ::
Faber      | ::                                       ::
Faber      | ::   - http                              ::
Faber      | ::   - https                             ::
Faber      | ::                                       ::
Faber      | :: Public DID Information:               ::
Faber      | ::                                       ::
Faber      | ::   - DID: VCzTt2b3Ts33ogFT3vsejR       ::
Faber      | ::                                       ::
Faber      | :: Administration API:                   ::
Faber      | ::                                       ::
Faber      | ::   - http://0.0.0.0:8021               ::
Faber      | ::                                       ::
Faber      | ::                          ver: 0.7.4-rc2 ::
Faber      | ::::::::::::::::::::::::::::::::::::::::::::
Faber      |
Faber      | Listening...
Faber      |
Startup duration: 3.53s
Admin URL is at: http://host.docker.internal:8021
```

```
Endpoint URL is at: http://host.docker.internal:8020

#3/4 Create a new schema/cred def on the ledger
Schema:
  {
    "sent": {
      "schema_id": "VCzTt2b3Ts33ogFT3vsejR:2:degree schema:44.94.65",
      "schema": {
        "ver": "1.0",
        "id": "VCzTt2b3Ts33ogFT3vsejR:2:degree schema:44.94.65",
        "name": "degree schema",
        "version": "44.94.65",
        "attrNames": [
          "date",
          "degree",
          "birthdate_dateint",
          "timestamp",
          "name"
        ],
        "seqNo": 8
      }
    },
    "schema_id": "VCzTt2b3Ts33ogFT3vsejR:2:degree schema:44.94.65",
    "schema": {
      "ver": "1.0",
      "id": "VCzTt2b3Ts33ogFT3vsejR:2:degree schema:44.94.65",
      "name": "degree schema",
      "version": "44.94.65",
      "attrNames": [
        "date",
        "degree",
        "birthdate_dateint",
        "timestamp",
        "name"
      ],
      "seqNo": 8
    }
  }

Schema ID: VCzTt2b3Ts33ogFT3vsejR:2:degree schema:44.94.65
Cred def ID: VCzTt2b3Ts33ogFT3vsejR:3:CL:8:faber.agent.degree_schema
Publish schema/cred def duration: 8.07s

#7 Create a connection to alice and print out the invite details
Generate invitation duration: 0.02s
Use the following JSON to accept the invite from another demo agent. Or use the QR
code to connect from a mobile agent.
Invitation Data:
{"@type": "https://didcomm.org/out-of-band/1.0/invitation", "@id": "0ff50f14-5292-
47e3-91dd-7086765d2341", "handshake_proto
cols": ["https://didcomm.org/didexchange/1.0"], "label": "faber.agent",
"services": [{"id": "#inline", "type": "did-communi
cation", "recipientKeys":
["did:key:z6MkkQEZESJsECzAARhR2SV4jt4XHXE5D5fwfiDzgKzEAuKf"], "serviceEndpoint":
```

```
"http://host.doc
ker.internal:8020"}]}
...
Waiting for connection...
```

Notons également que le code QR pour se connecter est également affiché lors du lancement.

Lancer l'Agent Alice

Depuis, un nouveau terminal nous allons lancer l'agent d'Alice :

```
Valon@WINDOWSDESKTOP MINGW64 ~/Desktop/Test/AriesPython/aries-cloudagent-
python/demo (main)
$ ./run_demo alice
Preparing agent image...
...
#7 Provision an agent and wallet, get back configuration details
Started webhook listener on port: 8032
Alice      | ['/home/indy/.pyenv/versions/3.6.13/bin/python', '-m',
'aries_cloudagent', 'start', '--endpoint',
'http://host.docker.internal:8030', '--label', 'alice.agent', '--auto-ping-
connection', '--auto-respond-message
s', '--inbound-transport', 'http', '0.0.0.0', '8030', '--outbound-transport',
'http', '--admin', '0.0.0.0', '80
31', '--admin-insecure-mode', '--wallet-type', 'indy', '--wallet-name',
'alice.agent361097', '--wallet-key', 'a
lice.agent361097', '--preserve-exchange-records', '--auto-provision', '--public-
invites', '--emit-new-didcomm-p
refix', '--genesis-transactions', '{"reqSignature":{},"txn":{"data":{"data":
{"alias":"Node1","blskey":"4N8aUNHS
gjQVgkpm8nhNEfDf6txHznoYREg9kirmJrkivgL4oSEimFF6nsQ6M41QvhM2Z33nves5vfSn9n1UwNFJBY
tWVnHYMATn76vLuL3zU88KyeAYcHf
sih3He6UHcXDxcaecHVz6jhCYz1P2UZn2bDVruL5wXpehgBfBaLKm3Ba","blskey_pop":"RahHYiCvoN
CtPTrVtP7nMC5eTYrsUA8WjXbdhNc
8debh1agE9bGiJxWBXYNFbnJXoXhWFMvyqhqhRoq737YQemH5ik9oL7R4NTTCz2LEZhkgLJzB3QRQqJyBN
yv7acbdHrAT8nQ9UkLbaVL9NBpnWX
BTw4LEMePaSHEw66RzPNdAX1","client_ip":"host.docker.internal","client_port":9702,"n
ode_ip":"host.docker.internal
","node_port":9701,"services":
["VALIDATOR"]},"dest":"Gw6pDLhcBcoQesN72qfotTgFa7cbuqZpkX3Xo6pLhPhv"},"metadata":
{"from":"Th7MpTaRZVRYnPiabds81Y"},"type":"0"},"txnMetadata":
{"seqNo":1,"txnId":"fea82e10e894419fe2bea7d96296a6d
46f50f93f9eeda954ec461b2ed2950b62"},"ver":"1"}\n{"reqSignature":{},"txn":{"data":
{"data":{"alias":"Node2","blsk
ey":"37rAPpXVoxzKhz7d9gkUe52XuXryuLXoM6P6LbWDB7LSbG62Lsb33sfG7zqS8TK1MXwuCHj1FKNzV
psnafmqLG1vXN88rt38mNFs9TENzm
4QHdBzsvCuoBnPH7rpYYDo9DZNJePaDvRvqJKByCabubJz3XXKbEeshzpz4Ma5QYpJqjk","blskey_pop
":"Qr658mWZ2YC8JXGXwMDQTzuZCW
F7NK9EwxphGmcBvCh6ybUuLxbG65nsX4JvD4SPNtkJ2w9ug1yLTj6fgmuDg41TgECXjLCij3RMsV8CwewB
VgVN67wsA45DFWvqvLtu4rjNnE9Jb
dFTc1Z4WCPA3Xan44K1HoHAq9EVeaRYs8zoF5","client_ip":"host.docker.internal","client_
port":9704,"node_ip":"host.do
cker.internal","node_port":9703,"services":
["VALIDATOR"]},"dest":"8ECVSk179mjsjKRLWiQtssMLgp6EPhWXtaYyStWPSGAb"
},"metadata":{"from":"EbP4aYNeTHL6q385GuVpRV"},"type":"0"},"txnMetadata":
{"seqNo":2,"txnId":"1ac8aece2a18ced660
fef8694b61aac3af08ba875ce3026a160acbc3a3af35fc"},"ver":"1"}\n{"reqSignature":
{},"txn":{"data":{"data":{"alias":
"Node3","blskey":"3WFpdbg7C5cnLYZwFZevJqhubkFALBfCBBok15GdrKMUhUjGsk3jV6QKj6MZgEub
F7oqCafxNdkm7eswgA4sdKTRc82tL
GzZBd6vNqU8dupzup6uYUf32KTHTPQbuUM8Yk4QFXjEf2Usu2TJcNkdgpyeUSX42u5LqdDDpNSWUK5deC5
","blskey_pop":"QwDeb2CkNSx6r
8QC8vGQK3GRv7Yndn84TGNijX8YXHPiagXajyfTjoR87rXUu4G4QLk2cF8NNyqWiYMus1623dELWwx57rL
CFqGh7N4ZRbGDRP4fnVcaKg1BcUxQ
```

```
866Ven4gw8y4N56S5HzxXNBZtLYmhGHvDtk6PFkFwCvxYrNYjh","client_ip":"host.docker.inter
nal","client_port":9706,"node
_ip":"host.docker.internal","node_port":9705,"services":
["VALIDATOR"]},"dest":"DKVxG2fXXTU8yT5N7hGEbXB3dfdAnYv1
JczDUHpmDxya"},"metadata":
{"from":"4cU41vWW82ArfxJxHkzXPG"},"type":"0"},"txnMetadata":
{"seqNo":3,"txnId":"7e9f3
55dffa78ed24668f0e0e369fd8c224076571c51e2ea8be5f26479edebe4"},"ver":"1"}\n{"reqSig
nature":{},"txn":{"data":{"da
ta":
{"alias":"Node4","blskey":"2zN3bHM1m4rLz54MJHYSwvqzPchYp8jkHswveCLAEJVcX6Mm1wHQD1S
kPYMzUDTZvWvhuE6VNAkK3KxV
eEmsanSmvjVkReDeBEMxeDaayjcZjFGPydyey1qxBHmTvAnBKoPydvuTAqx5f7YNNRAdeLmUi99gERUU7T
D8KfAa6MpQ9bw","blskey_pop":"
RPLagxaR5xdimFzwmzYnz4ZhWtYQEj8iR5ZU53T2gitPCyCHQneUn2Huc4oeLd2B2HzkGnjAff4hWTJT6C
7qHYB1Mv2wU5iHHGFWkhnTX9WsEAb
unJCV2qcaXScKj4tTfvdDKfLiVuU2av6hbsMztirRze7LvYBkRHV3tGwyCptsrP","client_ip":"host
.docker.internal","client_por
t":9708,"node_ip":"host.docker.internal","node_port":9707,"services":
["VALIDATOR"]},"dest":"4PS3EDQ3dW1tci1Bp65
43CfuuebjFrg36kLAUcskGfaA"},"metadata":
{"from":"TWwCRQRZ2ZHMJFn9TzLp7W"},"type":"0"},"txnMetadata":{"seqNo":4,"
txnId":"aa5e817d7cc626170eca175822029339a444eb0ee8f0bd20d3b0b76e566fb008"},"ver":"
1"}\n', '--webhook-url', 'htt
p://host.docker.internal:8032/webhooks', '--monitor-revocation-notification', '--
trace-target', 'log', '--trace
-tag', 'acapy.events', '--trace-label', 'alice.agent.trace', '--auto-accept-
invites', '--auto-accept-requests',
 '--auto-store-credential']
Alice      |
Alice      | ::::::::::::::::::::::::::::::::::::::::::::
Alice      | :: alice.agent                           ::
Alice      | ::                                       ::
Alice      | ::                                       ::
Alice      | :: Inbound Transports:                   ::
Alice      | ::                                       ::
Alice      | ::   - http://0.0.0.0:8030               ::
Alice      | ::                                       ::
Alice      | :: Outbound Transports:                  ::
Alice      | ::                                       ::
Alice      | ::   - http                              ::
Alice      | ::   - https                             ::
Alice      | ::                                       ::
Alice      | :: Administration API:                   ::
Alice      | ::                                       ::
Alice      | ::   - http://0.0.0.0:8031               ::
Alice      | ::                                       ::
Alice      | ::                          ver: 0.7.4-rc2 ::
Alice      | ::::::::::::::::::::::::::::::::::::::::::::
Alice      |
Alice      | Listening...
Alice      |
Startup duration: 3.02s
Admin URL is at: http://host.docker.internal:8031
```

```
Endpoint URL is at: http://host.docker.internal:8030

#9 Input faber.py invitation details
Invite details:
```

Connecter les deux agents

Nous pouvons voir que l'agent de Faber a généré des données d'invitation ensuite transformées également en QR code :


```
{
   "@type":"https://didcomm.org/out-of-band/1.0/invitation",
   "@id":"0ff50f14-5292-47e3-91dd-7086765d2341",
   "handshake_protocols":[
      "https://didcomm.org/didexchange/1.0"
   ],
   "label":"faber.agent",
   "services":[
      {
         "id":"#inline",
         "type":"did-communication",
         "recipientKeys":[
            "did:key:z6MkkQEZESJsECzAARhR2SV4jt4XHXE5D5fwfiDzgKzEAuKf"
         ],
         "serviceEndpoint":"http://host.docker.internal:8020"
      }
   ]
}
```


Pour se connecter depuis Alice, il suffit de copier l'invitation dans le BASH INPUT, puis les requêtes possibles apparaissent :


```
Invitation response:
  {
    "rfc23_state": "request-sent",
    "their_role": "inviter",
    "updated_at": "2022-05-17T13:30:46.963324Z",
    "state": "request",
    "invitation_key": "6wyWeC4RtfVh3vriLsXDtnWXTwxDoCRayhK4r42DFgYH",
    "invitation_msg_id": "0ff50f14-5292-47e3-91dd-7086765d2341",
    "request_id": "8e742659-0c64-4323-86e4-e86c1fdf6383",
    "connection_protocol": "didexchange/1.0",
    "invitation_mode": "once",
    "my_did": "UuSrYYtTvEr3B3RSmzi4zV",
    "their_label": "faber.agent",
    "accept": "auto",
    "connection_id": "033c0589-6012-48ef-bbfd-8c7f57277a10",
    "routing_state": "none",
    "created_at": "2022-05-17T13:30:46.928731Z"
```

```
  }

Connect duration: 0.06s
Waiting for connection...
Alice      | Connected
Alice      | Check for endorser role ...
Connect duration: 0.18s
    (3) Send Message
    (4) Input New Invitation
    (X) Exit?
[3/4/X]
```

L'agent de Faber affiche maintenant :

```
Faber      | Connected
Faber      | Check for endorser role ...
    (1) Issue Credential
    (2) Send Proof Request
    (2a) Send *Connectionless* Proof Request (requires a Mobile client)
    (3) Send Message
    (4) Create New Invitation
    (T) Toggle tracing on credential/proof exchange
    (X) Exit?
[1/2/3/4/T/X]
```

Envoyer des messages entre les deux agents

En utilisant l'option 3, nous pouvons envoyer des messages entre les deux agents. Voilà un exemple :

```
[1/2/3/4/T/X] 3
Enter message: Hello Alice
Faber      | Received message: alice.agent received your message
Faber      | Received message: Hello Faber
    (1) Issue Credential
    (2) Send Proof Request
    (2a) Send *Connectionless* Proof Request (requires a Mobile client)
    (3) Send Message
    (4) Create New Invitation
    (T) Toggle tracing on credential/proof exchange
    (X) Exit?
[1/2/3/4/T/X]
```

```
Alice      | Received message: Hello Alice
    (3) Send Message
    (4) Input New Invitation
    (X) Exit?
[3/4/X] 3
```

```
Enter message: Hello Faber
Alice      | Received message: faber.agent received your message
    (3) Send Message
    (4) Input New Invitation
    (X) Exit?
[3/4/X]
```

Délivrer et Prouver les Crédentiels

Pour tester, le protocole d'échange des crédentiels. Ainsi, nous allons aller sur l'agent de Faber pour envoyer (1) les crédentiels et demander une preuve (2) :

```
[1/2/3/4/T/X] 1

#13 Issue credential offer to X
Faber      | Credential: state = offer-sent, cred_ex_id = 6a3c1d2f-a812-4970-86f3-
a7b0bd8ef122
Faber      | Credential: state = request-received, cred_ex_id = 6a3c1d2f-a812-
4970-86f3-a7b0bd8ef1
22

#17 Issue credential to X
Faber      | Credential: state = credential-issued, cred_ex_id = 6a3c1d2f-a812-
4970-86f3-a7b0bd8ef
122
Faber      | Credential: state = done, cred_ex_id = 6a3c1d2f-a812-4970-86f3-
a7b0bd8ef122
    (1) Issue Credential
    (2) Send Proof Request
    (2a) Send *Connectionless* Proof Request (requires a Mobile client)
    (3) Send Message
    (4) Create New Invitation
    (T) Toggle tracing on credential/proof exchange
    (X) Exit?
[1/2/3/4/T/X] 2

#20 Request proof of degree from alice
Faber      | Presentation: state = request-sent, pres_ex_id = 118ee324-d47c-495f-
a03f-e490d3177182
Faber      | Presentation: state = presentation-received, pres_ex_id = 118ee324-
d47c-495f-a03f-e49
0d3177182

#27 Process the proof provided by X

#28 Check if proof is valid
Faber      | Presentation: state = done, pres_ex_id = 118ee324-d47c-495f-a03f-
e490d3177182
Faber      | Proof = true
    (1) Issue Credential
    (2) Send Proof Request
```

```
    (2a) Send *Connectionless* Proof Request (requires a Mobile client)
    (3) Send Message
    (4) Create New Invitation
    (T) Toggle tracing on credential/proof exchange
    (X) Exit?
[1/2/3/4/T/X]
```

Il n'y a rien besoin de faire avec l'agent d'Alice car il est implémenté pour automatiquement recevoir des crédentiels et répondre à une requête de preuve.

```
Alice      | Credential: state = offer-received, cred_ex_id = 2d52de5c-8566-46f7-
a98b-7153bd1fd84a

#15 After receiving credential offer, send credential request
Alice      | Credential: state = request-sent, cred_ex_id = 2d52de5c-8566-46f7-
a98b-7153bd1fd84a
Alice      | Credential: state = credential-received, cred_ex_id = 2d52de5c-8566-
46f7-a98b-7153bd1
fd84a

#18.1 Stored credential 99cb65f8-77c1-4df5-86c8-63f10867a07e in wallet
Credential details:
  {
    "referent": "99cb65f8-77c1-4df5-86c8-63f10867a07e",
    "attrs": {
      "birthdate_dateint": "19980517",
      "timestamp": "1652795774",
      "date": "2018-05-28",
      "degree": "Maths",
      "name": "Alice Smith"
    },
    "schema_id": "VCzTt2b3Ts33ogFT3vsejR:2:degree schema:44.94.65",
    "cred_def_id": "VCzTt2b3Ts33ogFT3vsejR:3:CL:8:faber.agent.degree_schema",
    "rev_reg_id": null,
    "cred_rev_id": null
  }

Alice      | credential_id 99cb65f8-77c1-4df5-86c8-63f10867a07e
Alice      | cred_def_id VCzTt2b3Ts33ogFT3vsejR:3:CL:8:faber.agent.degree_schema
Alice      | schema_id VCzTt2b3Ts33ogFT3vsejR:2:degree schema:44.94.65
Alice      | Credential: state = done, cred_ex_id = 2d52de5c-8566-46f7-a98b-
7153bd1fd84a
Alice      | Presentation: state = request-received, pres_ex_id = 43ea061d-b06a-
4ba1-a889-ac94e698
82df

#24 Query for credentials in the wallet that satisfy the proof request

#25 Generate the proof

#26 Send the proof to X: {"indy": {"requested_predicates":
```

```
{"0_birthdate_dateint_GE_uuid": {"cred_
id": "99cb65f8-77c1-4df5-86c8-63f10867a07e"}}, "requested_attributes":
{"0_name_uuid": {"cred_id":
 "99cb65f8-77c1-4df5-86c8-63f10867a07e", "revealed": true}, "0_date_uuid":
{"cred_id": "99cb65f8-7
7c1-4df5-86c8-63f10867a07e", "revealed": true}, "0_degree_uuid": {"cred_id":
"99cb65f8-77c1-4df5-8
6c8-63f10867a07e", "revealed": true}}, "self_attested_attributes": {}}}
Alice      | Presentation: state = presentation-sent, pres_ex_id = 43ea061d-b06a-
4ba1-a889-ac94e69
882df
Alice      | Presentation: state = done, pres_ex_id = 43ea061d-b06a-4ba1-a889-
ac94e69882df
    (3) Send Message
    (4) Input New Invitation
    (X) Exit?
[3/4/X]
```

Plus de possibilités

Nous pouvons trouver de plus amples informations sur les possibilités de la démo via ce lien.

Code à disposition

Il y a trois fichiers intéressants qui, avec les explications du tutoriel, nous permettent d'un peu mieux comprendre le fonctionnement du code. Ces fichiers sont les suivants :

- agent.py
- alice.py
- faber.py

HyperLedger Aries - Apprendre l'utilisation de l'API grâce à Swagger

Ce document montre la reproduction du lab suivant sur une machine wsl2 Ubuntu sur Windows 11.

Prérequis

Pour effectuer ce Lab, il est nécessaire d'avoir :

- Git
- Docker

Avant-Propos

Le lancement d'ACA-Py génère automatiquement une configuration OpenAPI. Grâce à cela, il est possible d'avoir une vision sur tous les appels exposés par ACA-Py, avec des exemples et la possibilité de les essayer. De plus, grâce à la définition, il est possible de générer du code dans notre langage de préférence qui servira de squelette au contrôleur sans rien avoir besoin de coder jusque-là.

Étapes de réalisation

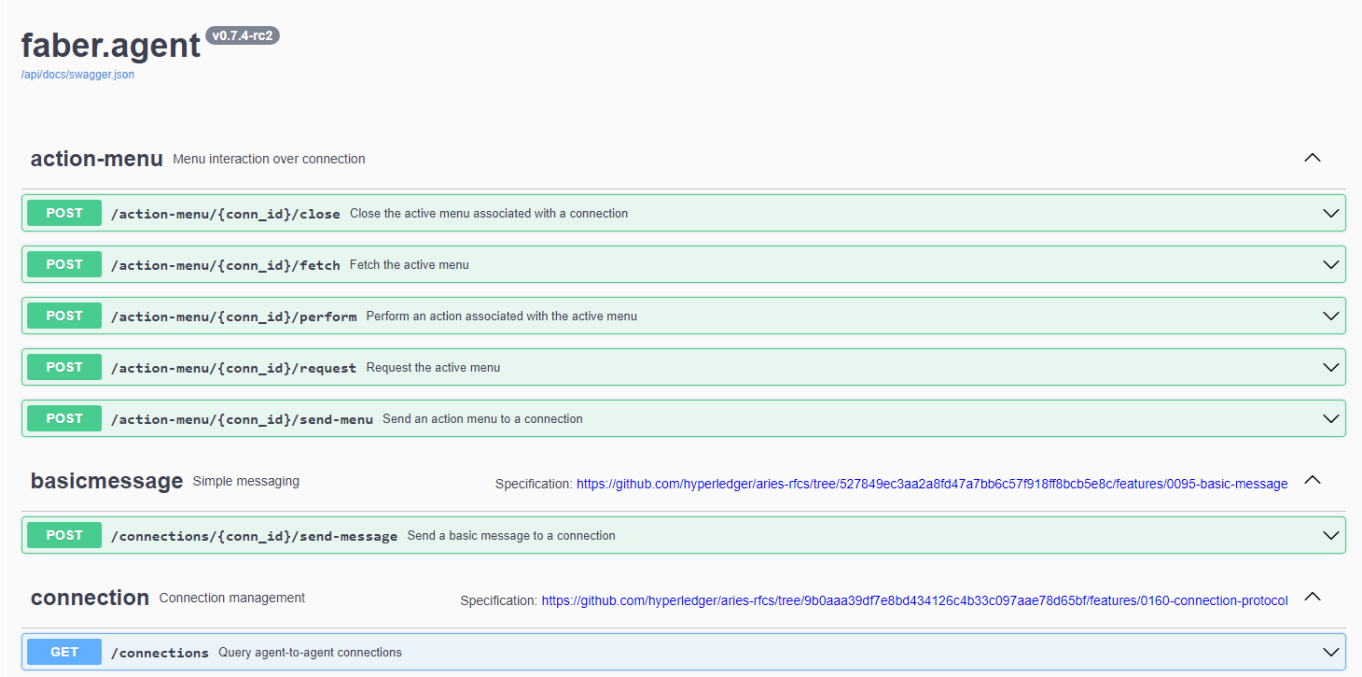
Nous partons du principe que le github suivant a déjà été cloner sur la machine.

Lancement de l'Agent de Faber avec un LEDGER_URL personnalisé

Nous allons donc tout d'abord démarrer l'agent de Faber avec comme Ledger personnalisé l'url suivant. Pour rappel, n'importe quel url de leger peut être utilisé tant que {url/genesis} conduit vers le fichier genesis du ledger, par exemple :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython$ cd aries-cloudagent-python/demo/
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-cloudagent-python/demo$
LEDGER_URL=http://dev.greenlight.bcovrin.vonx.io ./run_demo faber
Preparing agent image...
...
...
Waiting for connection...
```

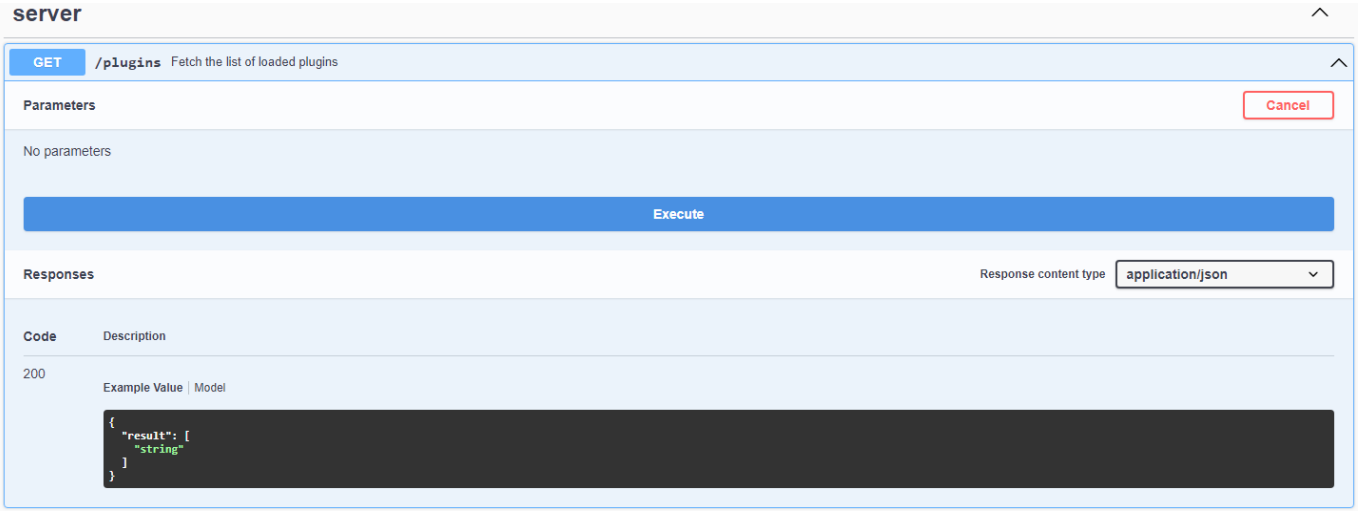
L'instance de l'Agent de Faber est maintenant lancée, mais cette fois nous n'allons pas jouer avec les lignes de commandes, mais nous allons nous rendre sur l'adresse http://localhost:8021/ pour visualiser l'interface de l'API de l'instance Faber que nous avons lancé. Sur ce lien, une longue liste d'actions HTTP s'affiche. Voilà un échantillon :

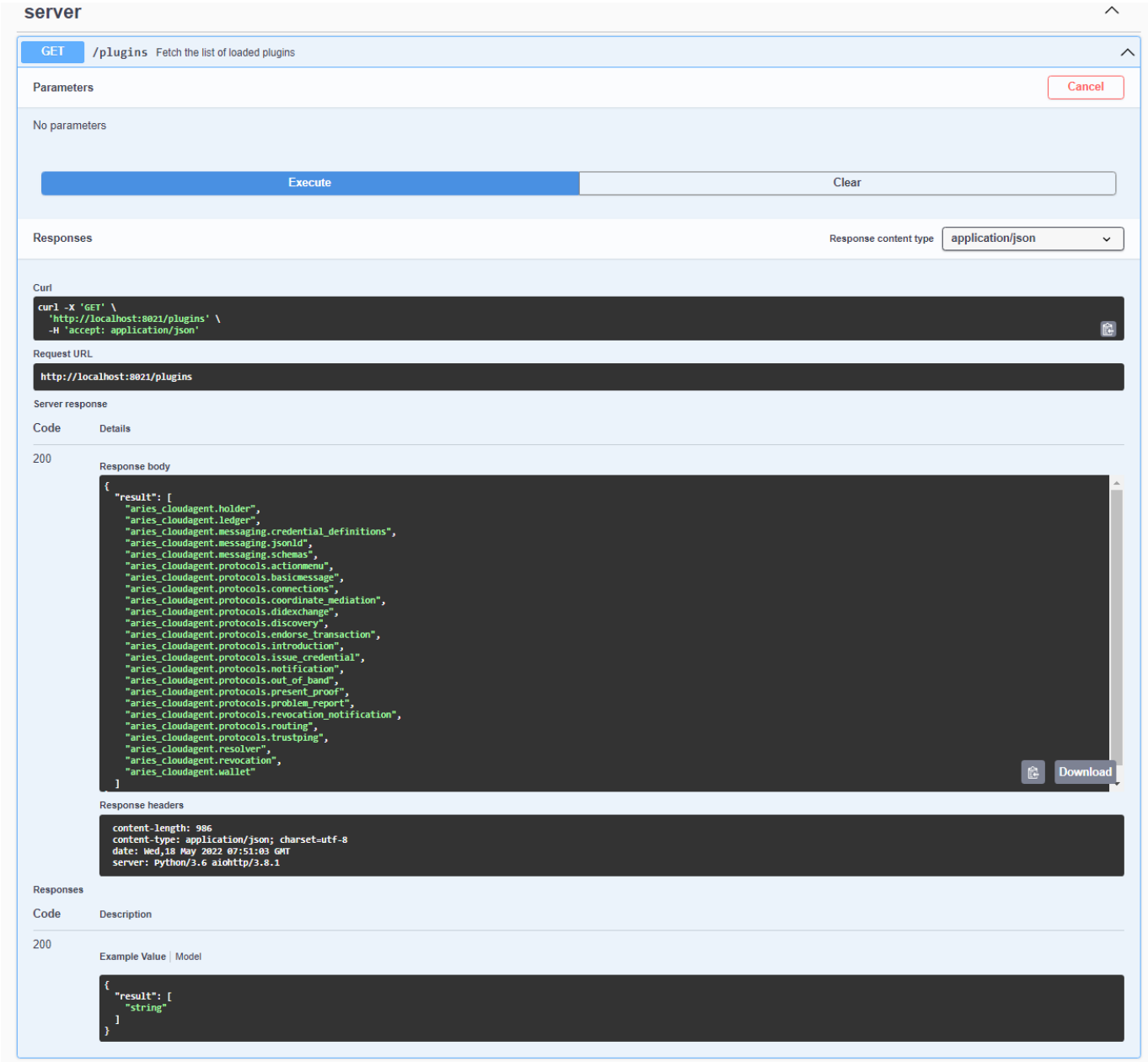


Utilisation de l'OpenAPI

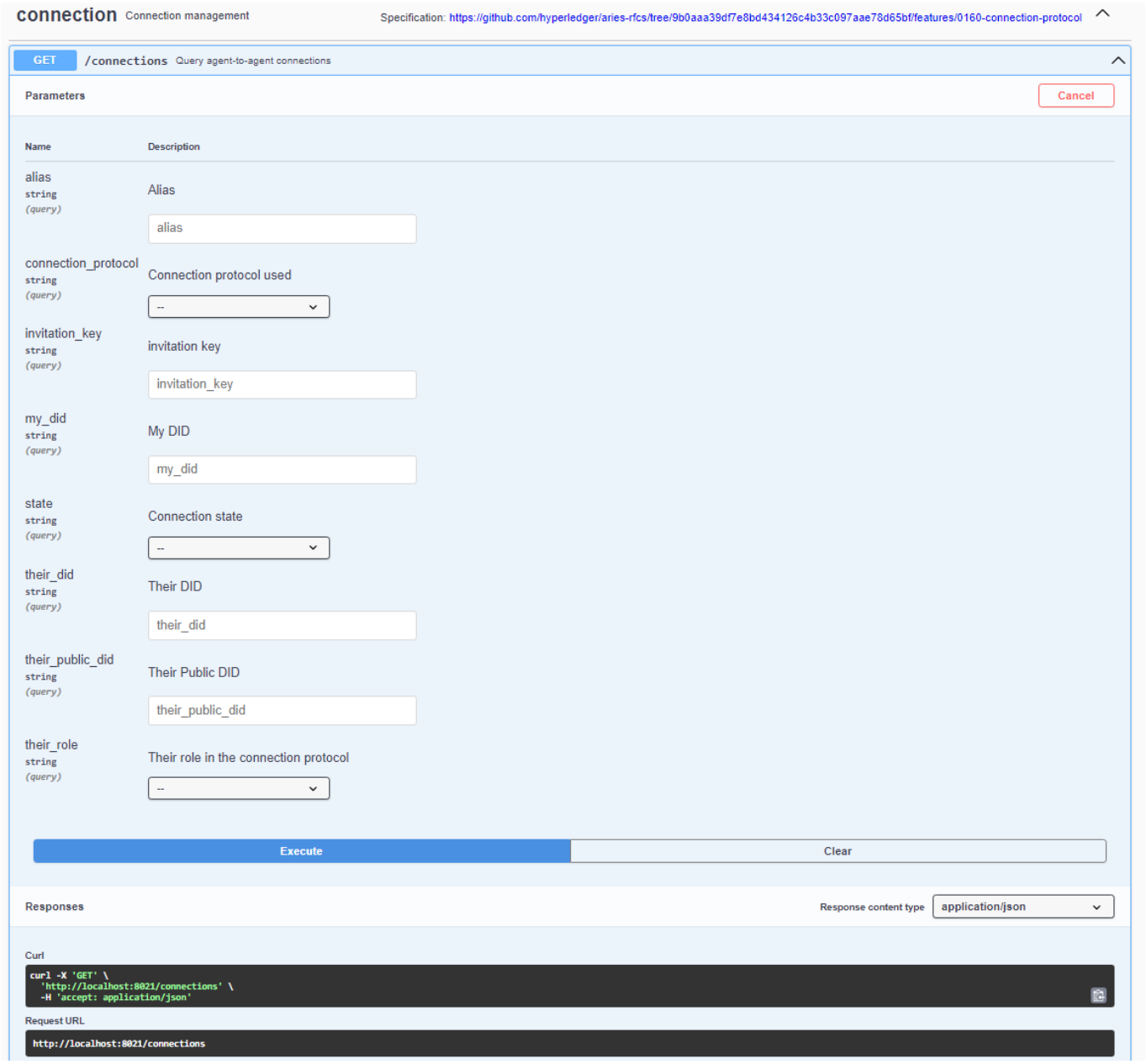
Nous pouvons tester l'API en faisant des requêtes depuis cette interface. Ce qui est intéressant, c'est que celle-ci est très détaillée et permet d'avoir rapidement une vue sur les exemples d'utilisation, sur les protocoles utilisés etc...

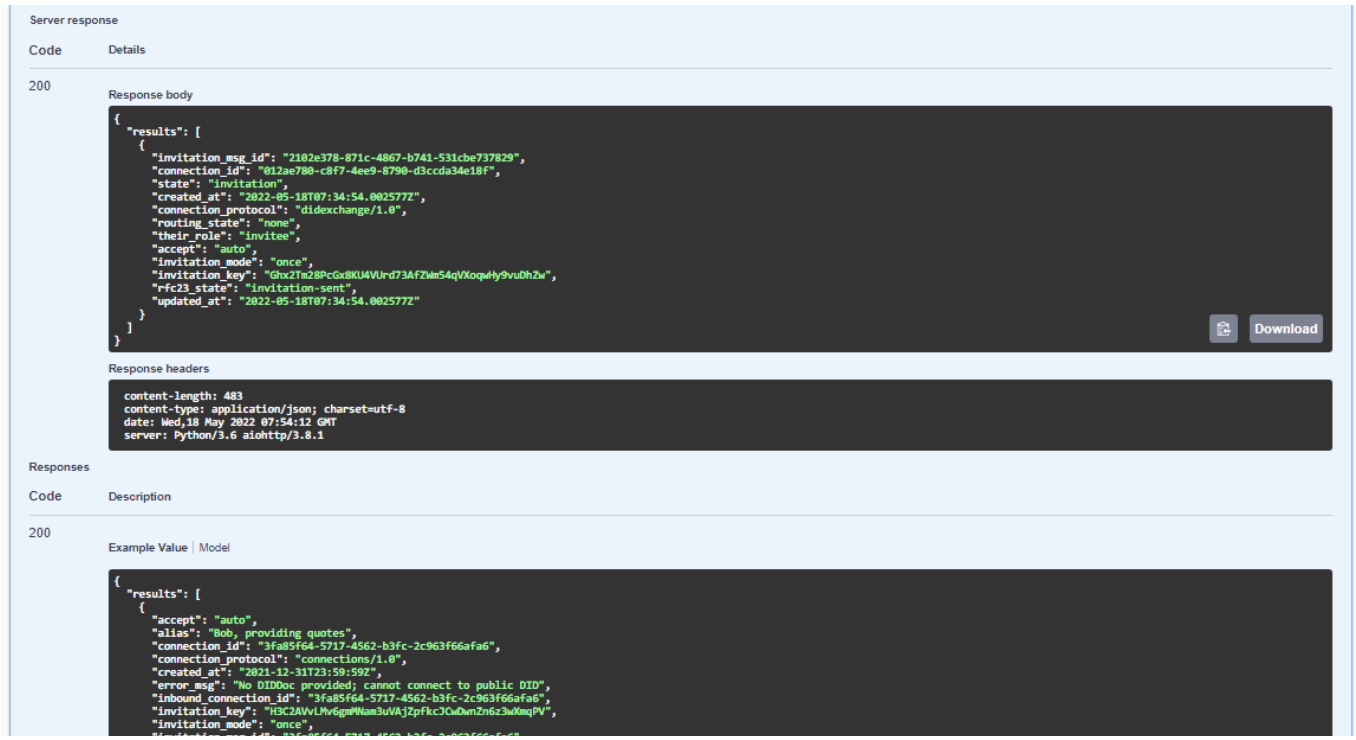
Prenons par exemple l'exécution de la méthode GET "/plugin" dans la section "server". Pour l'exécuter, il suffit d'ouvrir la requête, de presser le bouton "execute" et nous obtenons la réponse du serveur, qui est la liste des plugins utilisés dans notre instance.





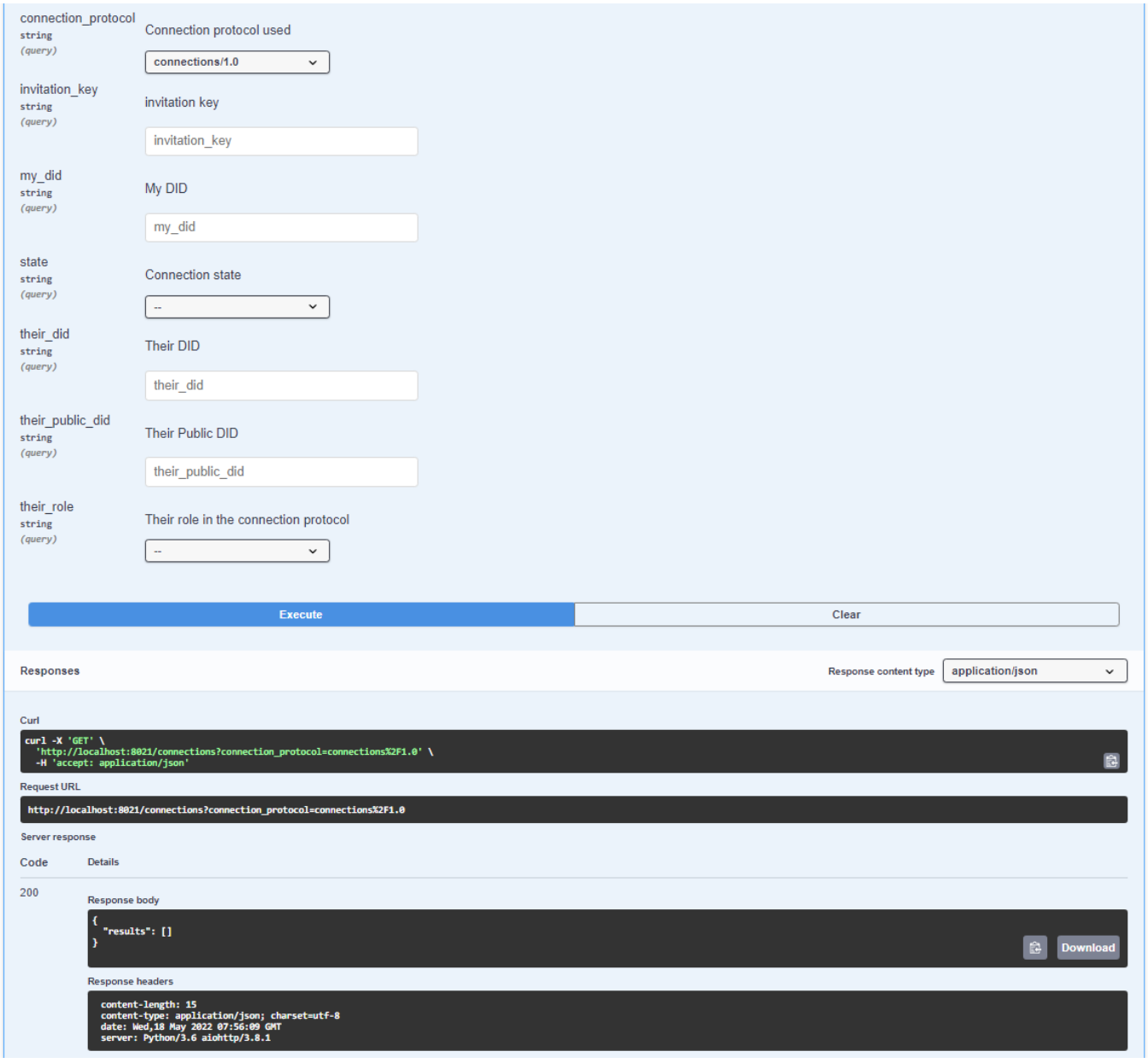
Il y a bien évidemment des actions qui prennent des variables en paramètres. Intéressons-nous à la méthode GET "/connections" et essayons de l'exécuter sans aucun paramètre :


connection Connection management
Specification: https://github.com/hyperledger/aries-rfcs/tree/9b0aaa39df7e8bd434126c4b33c097aae78d65bf/features/0160-connection-protocol
GET /connections Query agent-to-agent connections
Parameters
Cancel
Name
Description
alias
string
(query)
Alias
alias
connection_protocol
string
(query)
Connection protocol used
--
invitation_key
string
(query)
invitation key
invitation_key
my_did
string
(query)
My DID
my_did
state
string
(query)
Connection state
--
their_did
string
(query)
Their DID
their_did
their_public_did
string
(query)
Their Public DID
their_public_did
their_role
string
(query)
Their role in the connection protocol
--
Execute
Clear
Responses
Response content type
application/json
Curl
curl -X 'GET' \
'http://localhost:8021/connections' \
-H 'accept: application/json'
Request URL
http://localhost:8021/connections


Server response
Code
Details
200
Response body
{
"results": [
{
"invitation_msg_id": "2102e378-871c-4867-b741-531cbe737829",
"connection_id": "012ae780-c8f7-4ee9-8790-d3ccda34e18f",
"state": "invitation",
"created_at": "2022-05-18T07:34:54.002577Z",
"connection_protocol": "didexchange/1.0",
"routing_state": "none",
"their_role": "invitee",
"accept": "auto",
"invitation_mode": "once",
"invitation_key": "Ghx2Tm28PcGx8KU4VUrd73AfZWm54qVXoqw4y9vuDhZw",
"rfc23_state": "invitation-sent",
"updated_at": "2022-05-18T07:34:54.002577Z"
}
]
}
Download
Response headers
content-length: 483
content-type: application/json; charset=utf-8
date: Wed,18 May 2022 07:54:12 GMT
server: Python/3.6 aiohttp/3.8.1
Responses
Code
Description
200
Example Value | Model
{
"results": [
{
"accept": "auto",
"alias": "Bob, providing quotes",
"connection_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
"connection_protocol": "connections/1.0",
"created_at": "2021-12-31T23:59:59Z",
"error_msg": "No DIDDoc provided; cannot connect to public DID",
"inbound_connection_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
"invitation_key": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV",
"invitation_mode": "once",

```
      "invitation_msg_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
      "my_did": "WgWxqztrNooG92RXvxSTWv",
      "request_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
      "rfc23_state": "invitation-sent",
      "routing_state": "active",
      "state": "active",
      "their_did": "WgWxqztrNooG92RXvxSTWv",
      "their_label": "Bob",
      "their_public_did": "2cpBmR3FqGKWi5EyUbpRY8",
      "their_role": "requester",
      "updated_at": "2021-12-31T23:59:59Z"
    }
  ]
}
```


Nous pouvons voir dans le résultat que nous avons qu'une seule connexion active. Ainsi maintenant, si dans la liste des paramètres nous spécifions par exemple en "connection_protocol" la valeur "connections/1.0", nous n'aurons aucun résultat :

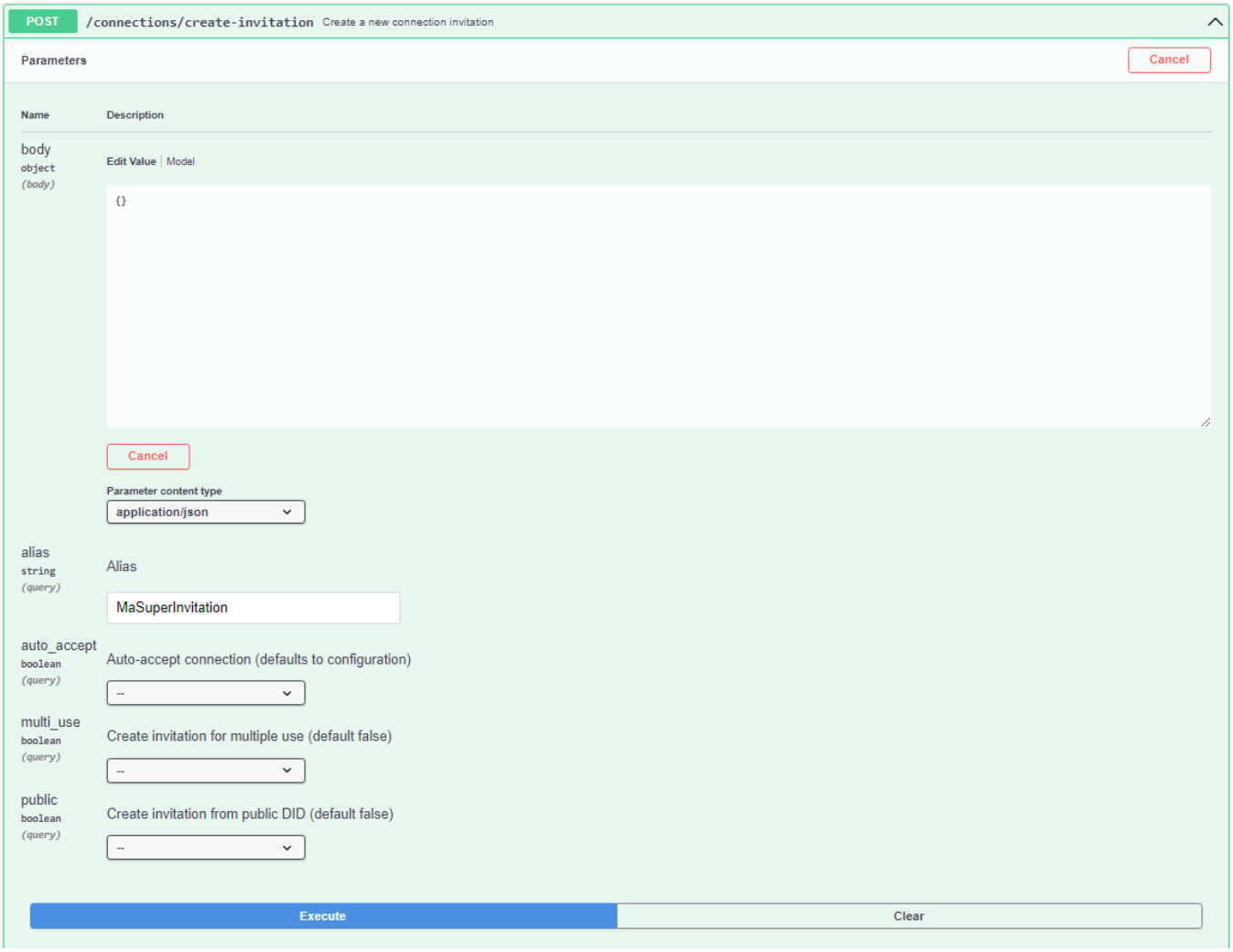


Bien évidemment, si nous utilisions la valeur "didexchange/1.0", le même résultat qu'au début apparaîtra vu qu'il correspond au paramètre de la requête.

Exemple

Nous allons maintenant créer une invitation et l'utiliser pour créer une connexion de Faber à lui-même.

Tout d'abord, il faut créer une invitation en utilisant la méthode POST "/connections/create-invitation" :



Curl

```
curl -X 'POST' \
  'http://localhost:8021/connections/create-invitation?alias=MaSuperInvitation' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{}'
```

Request URL

```
http://localhost:8021/connections/create-invitation?alias=MaSuperInvitation
```

Server response

Code Details

200

Response body


```
{
  "connection_id": "0a31bef5-78f5-4101-afa3-8df1ece16953",
  "invitation": {
    "@type": "https://didcomm.org/connections/1.0/invitation",
    "@id": "ed546aa5-390d-4969-a65a-bd4762b539c1",
    "serviceEndpoint": "http://192.168.65.3:8020",
    "recipientKeys": [
      "5JVgqEP5kiYTBm58QkHGxsZUnDmNDQeJmpStyVpt2qmn"
    ],
    "label": "faber.agent"
  },
  "invitation_url": "http://192.168.65.3:8020?c_i=eyJAdHlwZSI6ICJodHRwczovL2RpZGNvbW0ub3JnL2Nvbm5lY3Rpb25zLzEuMC9pbnZpdGF0aW9uIiwgIkBpZCI6ICJlZDU0NmFhNS0zOTBkLTQ5NjktYTY1YS1iZDQ3NjIxNTM5YzEiLCAic2VydmljZUVuZHBvaW50IjogImh0dHA6Ly8xOTIuMTY4LjY1LjM6ODAyMCIsICJyZWNpcGllbnRLZXlzIjogWyI1Sl2ncUVQNWtpWVRCbTU4UWtIR3hzWlVuRG1ORFFlSmiwU3R5VnB0MnFtbiJdLCAibGFiZWwiOiAiZmFiZXIuYWdlbnQifQ==",
  "alias": "MaSuperInvitation"
}
```


Download

Response headers

```
access-control-allow-credentials: true
access-control-allow-origin: http://localhost:8021
access-control-expose-headers: Server,Date,Content-Length
content-length: 726
content-type: application/json; charset=utf-8
date: Wed,18 May 2022 08:13:04 GMT
server: Python/3.6 aiohttp/3.8.1
```

Responses

Code Description

200

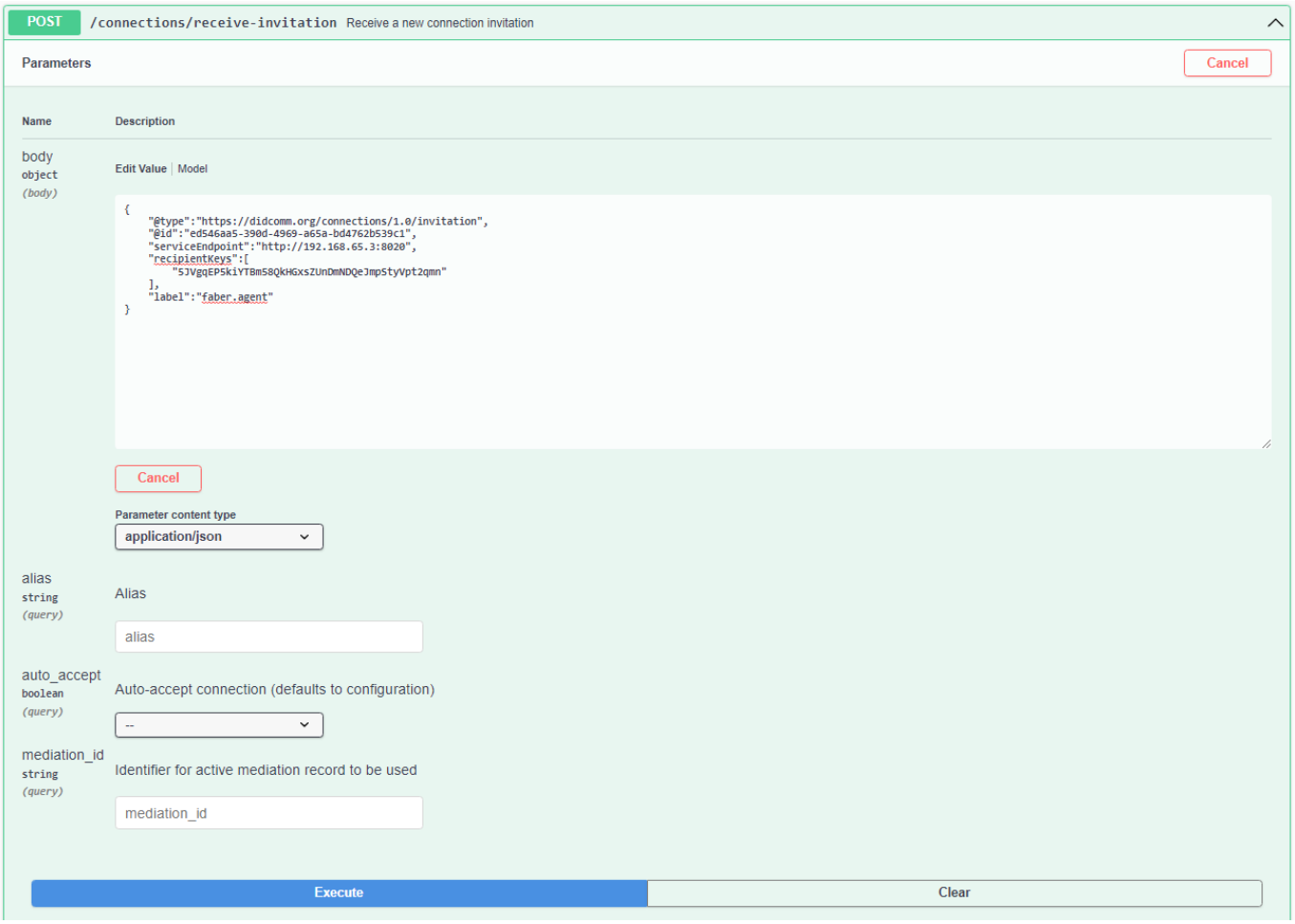
Example Value | Model

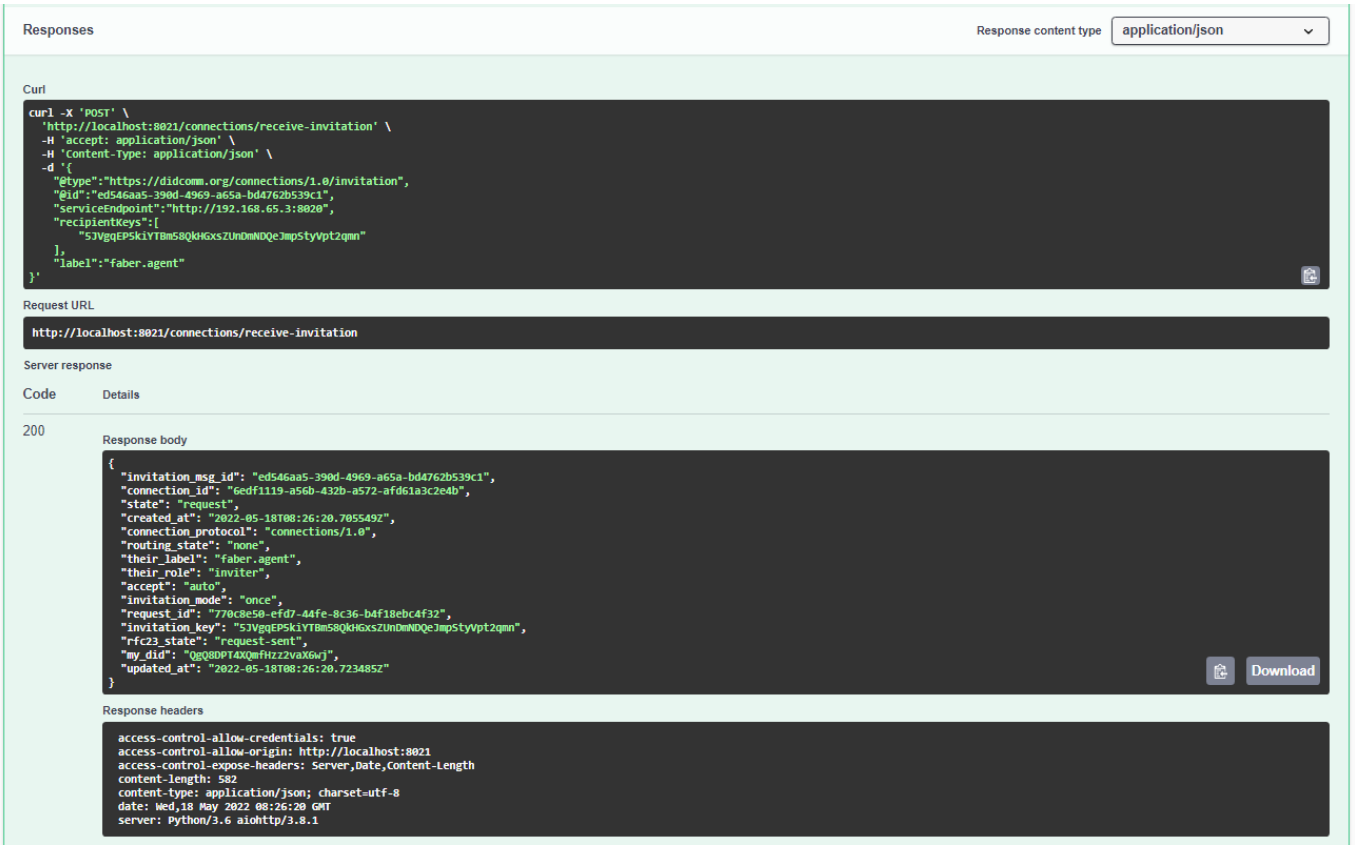

```
{
  "connection_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "invitation": {
    "@id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "@type": "https://didcomm.org/my-family/1.0/my-message-type",
    "did": "WgWxqztrNooG92RXvxSTWv",
    "imageUrl": "http://192.168.56.101/img/logo.jpg",
    "label": "Bob",
    "recipientKeys": [
      "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
    ],
    "routingKeys": [
      "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
    ],
    "serviceEndpoint": "http://192.168.56.101:8020"
  },
```

Nous avons donc maintenant une réponse qui contient l'invitation. Pour la suite, nous allons copier la valeur de l'élément "invitation" de { à } :

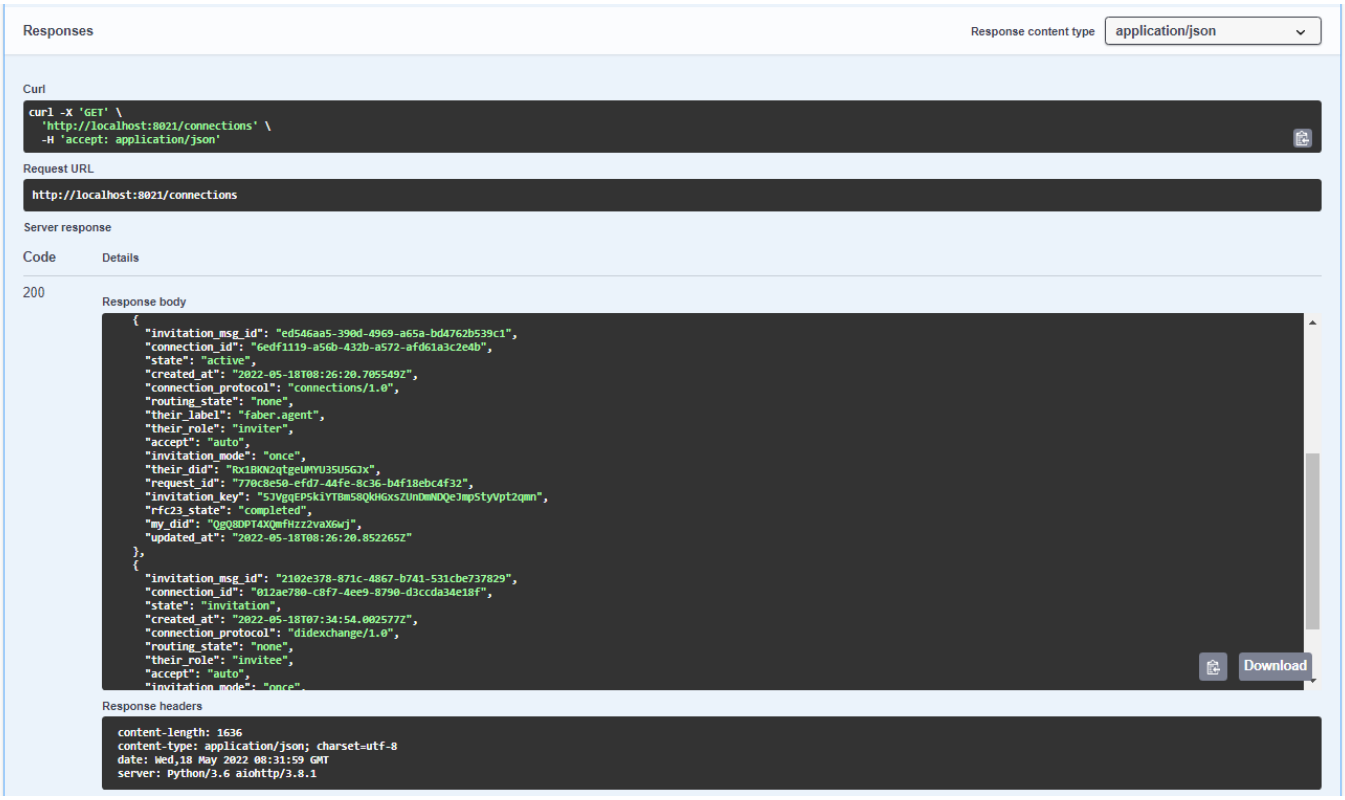

```
{
    "@type":"https://didcomm.org/connections/1.0/invitation",
    "@id":"ed546aa5-390d-4969-a65a-bd4762b539c1",
    "serviceEndpoint":"http://192.168.65.3:8020",
    "recipientKeys":[
        "5JVgqEP5kiYTBm58QkHGxsZUnDmNDQeJmpStyVpt2qmn"
    ],
    "label":"faber.agent"
}
```


Maintenant grâce à la méthode POST "/connections/receive-invitation", nous pouvons accepter l'invitation en copiant dans le "body" la valeur de l'invitation que nous venons de créer :

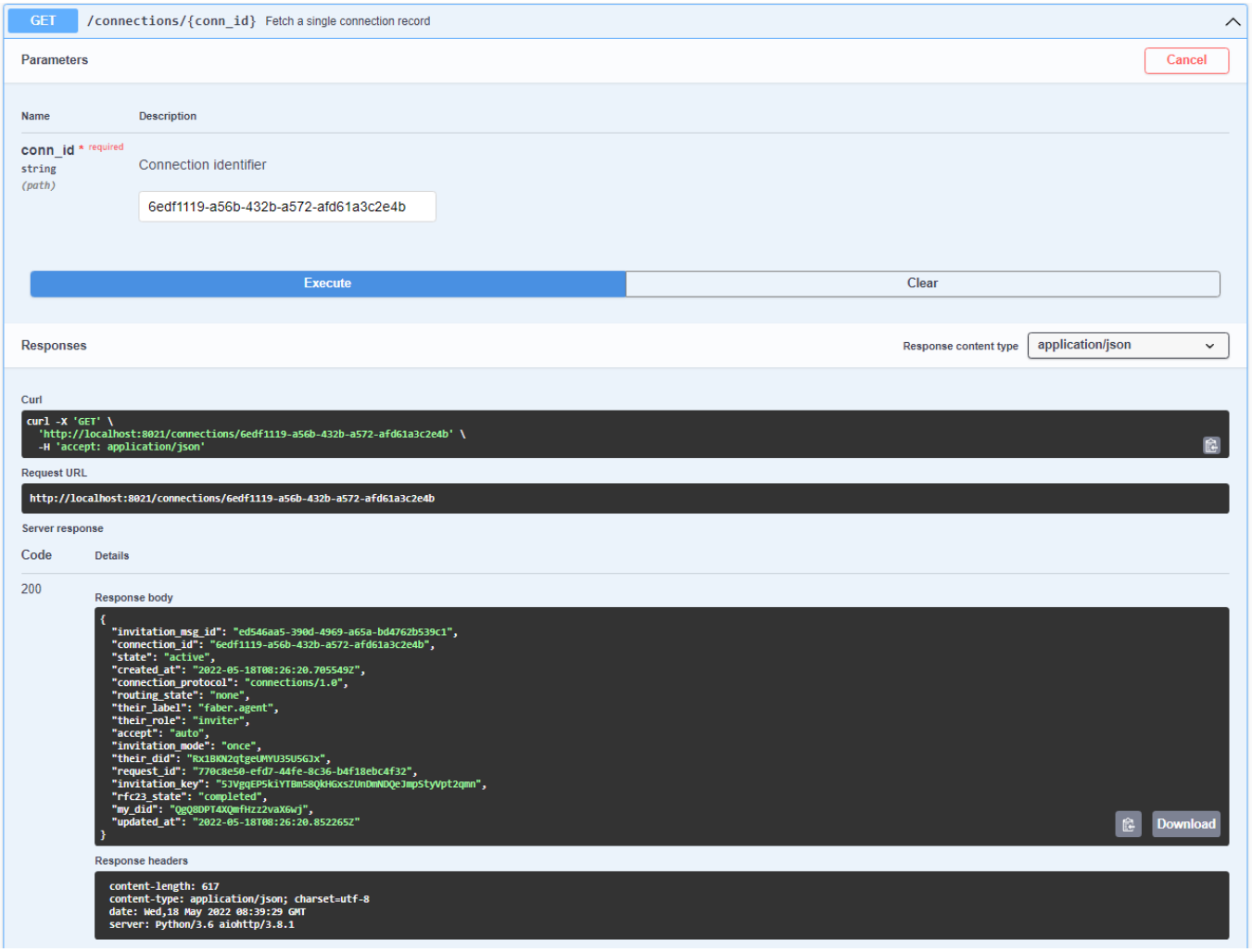




Nous pouvons voir que le "state" de l'invitation est en "request". Allons maintenant obtenir une liste des connexions avec la méthode GET "/connections" :



Si nous copions le champ "connection_id" `"6edf1119-a56b-432b-a572-afd61a3c2e4b"` et que nous le copions dans le champ "id" de la méthode GET "/connections/{id}", nous obtenons les détails de la connexion active :


GET /connections/{conn_id} Fetch a single connection record
Parameters
Cancel
Name
Description
conn_id * required
string
(path)
Connection identifier
6edf1119-a56b-432b-a572-afd61a3c2e4b
Execute
Clear
Responses
Response content type
application/json
Curl
curl -X 'GET' \
'http://localhost:8021/connections/6edf1119-a56b-432b-a572-afd61a3c2e4b' \
-H 'accept: application/json'
Request URL
http://localhost:8021/connections/6edf1119-a56b-432b-a572-afd61a3c2e4b
Server response
Code
Details
200
Response body
{
"invitation_msg_id": "ed546aa5-390d-4969-a65a-bd4762b539c1",
"connection_id": "6edf1119-a56b-432b-a572-afd61a3c2e4b",
"state": "active",
"created_at": "2022-05-18T08:26:20.705549Z",
"connection_protocol": "connections/1.0",
"routing_state": "none",
"their_label": "faber.agent",
"their_role": "inviter",
"accept": "auto",
"invitation_mode": "once",
"their_did": "Rx1BKN2qtgeUMYU35U5GJx",
"request_id": "770c8e50-efd7-44fe-8c36-b4f18ebc4f32",
"invitation_key": "5JVgqEP5kiYTBm58QkHGxsZUnDmNDQe3mpStyVpt2qmn",
"rfc23_state": "completed",
"my_did": "QgQ8DPT4XQmfHzz2vaX6Wj",
"updated_at": "2022-05-18T08:26:20.852265Z"
}
Download
Response headers
content-length: 617
content-type: application/json; charset=utf-8
date: Wed,18 May 2022 08:39:29 GMT
server: Python/3.6 aiohttp/3.8.1

Hyperledger Aries - Développement d'un Controller en Python

Ce document montre la reproduction du lab suivant sur une machine wsl2 Ubuntu sur Windows 11.

Prérequis

Pour effectuer ce Lab, il est nécessaire d'avoir :

- Docker

Situation Désirée

Dans ce lab, nous voulons atteindre la situation suivante. Faber doit envoyer des crédentiels à Alice pour que celle-ci puisse prouver au contrôleur de l'ACME CORP (une entreprise) qu'elle possède bien un diplôme. Si c'est le cas, l'acme envoie à Alice un crédentiel de travail. Nous allons donc compléter le code du code de acme.py.

Étapes de réalisation

Nous partons du principe que le github suivant a déjà été cloner sur la machine.

Nous allons donc modifier le fichier "aries-cloudagent-python/demo/runners/acme.py".

Demande de Proof

Commençons par ajouter des "import" que nous allons utiliser dans la classe :

```
import random

from datetime import date
from uuid import uuid4
```

Puis nous définissons/modifions les variables suivantes :

```
TAILS_FILE_COUNT = int(os.getenv("TAILS_FILE_COUNT", 100))
CRED_PREVIEW_TYPE = (
    "did:sov:BzCbsNYhMrjHiqZDTUASHg;spec/issue-credential/2.0/credential-preview"
)
```

Dans le fonction main(), nous localisons le code qui sera lancé lorsque l'option "2" est choisie :

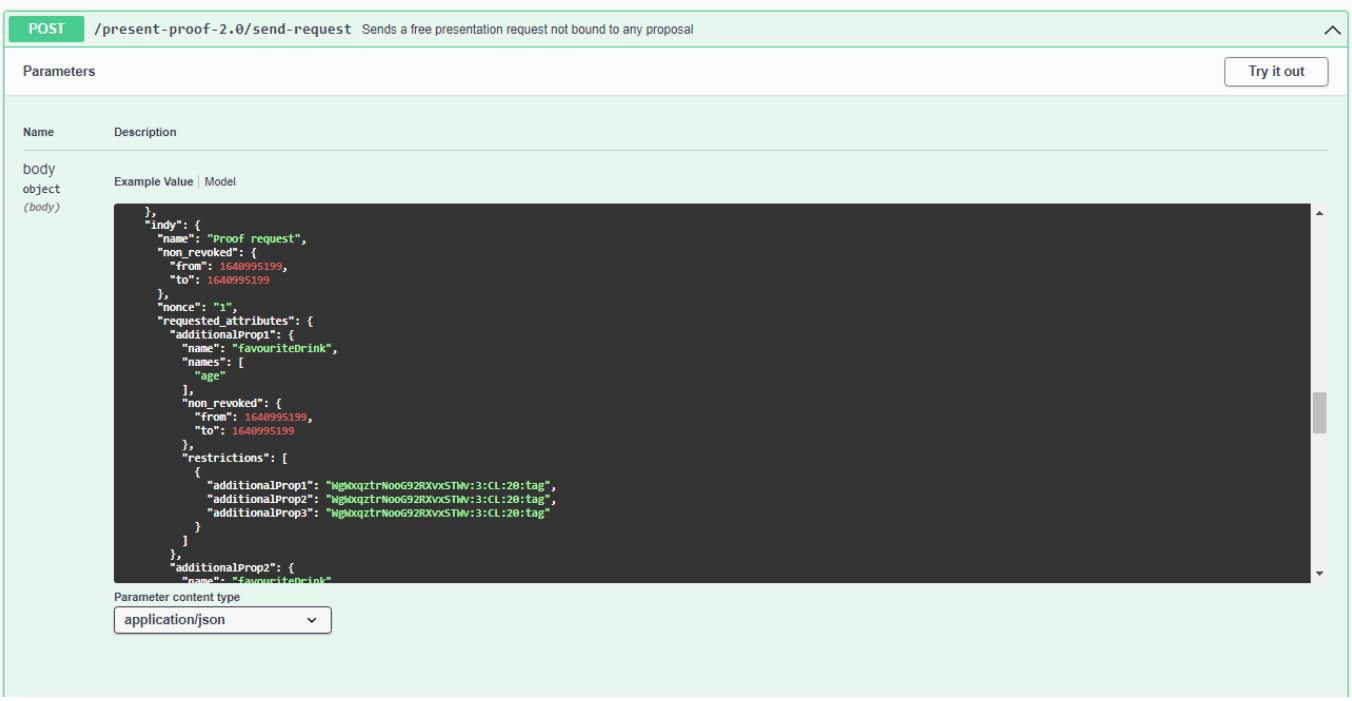
```
elif option == "2":
    log_status("#20 Request proof of degree from alice")
```

```
    # TODO presentation requests
```

Nous allons développer la partie TODO. Le code complet est le suivant :

```
elif option == "2":
    log_status("#20 Request proof of degree from alice")
    req_attrs = [
        {
            "name": "name",
            "restrictions": [{"schema_name": "degree schema"}]
        },
        {
            "name": "date",
            "restrictions": [{"schema_name": "degree schema"}]
        },
        {
            "name": "degree",
            "restrictions": [{"schema_name": "degree schema"}]
        }
    ]
    indy_proof_request = {
        "name": "Proof of Education",
        "version": "1.0",
        "nonce": str(uuid4().int),
        "requested_attributes": {
            f"0_{req_attr['name']}_uuid": req_attr for req_attr in req_attrs
        },
        "requested_predicates": {}
    }
    proof_request_web_request = {
        "connection_id": agent.connection_id,
        "presentation_request": {"indy": indy_proof_request},
    }
    # cela va envoyer la requête à l'agent, qui lui va s'occuper de
    # l'envoyer à Alice en se basant sur le connection_id
    await agent.admin_POST(
        "/present-proof-2.0/send-request",
        proof_request_web_request
    )
```

Ce que ce code fait est de simplement créer une requête POST sur la méthode "/present-proof-2.0/send-request" de l'API. Si nous utilisons l'API Swagger de Faber du lab5 pour voir à quoi correspond cette méthode, nous avons un modèle attendu par la méthode :



Nous voyons dans cette partie du body, la structure de l'objet "proof_request_web_request" que nous créons dans le code. Au début du code, nous définissons également ce que nous attendons comme attributs avec son nom et ses restrictions. Si nous fouillons un peu dans le code, dans le fichier "aries-cloudagent-python/demo/runners/agent_container.py" Faber initialise un schéma "degree schema" avec les attributs que nous utilisons dans le acme.py :

```
await faber_container.initialize(
            schema_name="degree schema",
            schema_attrs=[
                "name",
                "date",
                "degree",
                "grade",
            ],
        )
```

Réception de la proof

Nous allons maintenant développer la gestion lors de la réception de la proof. Pour ce faire, il faut se rendre dans la méthode handle_present_proof_v2_0 :

```
async def handle_present_proof_v2_0(self, message):
    state = message["state"]
    pres_ex_id = message["pres_ex_id"]
    self.log(f"Presentation: state = {state}, pres_ex_id = {pres_ex_id}")

    if state == "presentation-received":
        # TODO handle received presentations
        pass
```

Cette action interviendra après l'appel d'Alice à la méthode POST "/present-proof-2.0/records/{pres_ex_id}/send-presentation". Le paramètre pres_ex_id aussi utilisé lors de la récupération du message, correspond au "Presentation Exchange Identifier".

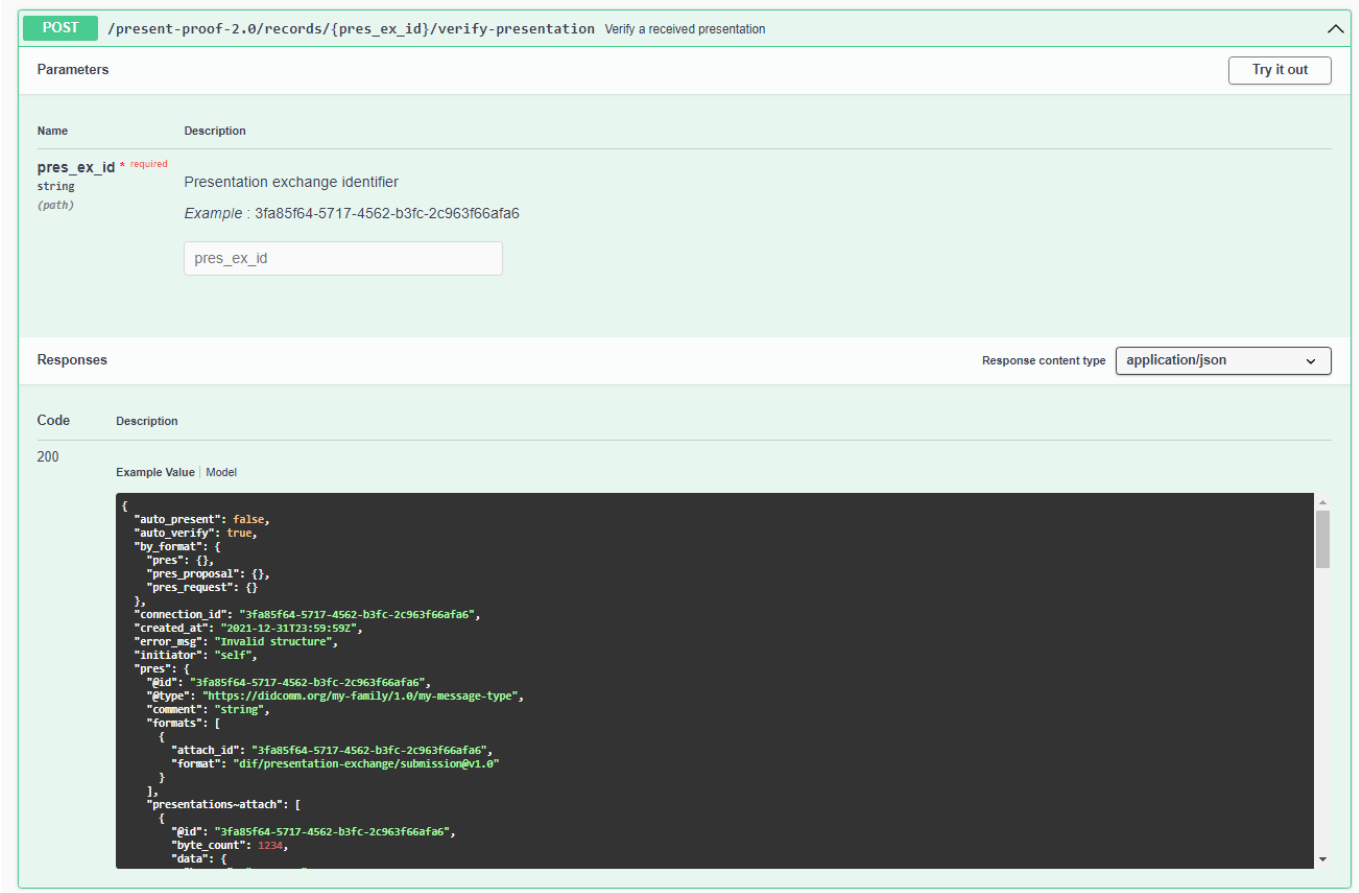
Voilà donc la version finale de la méthode handle_present_proof_v2_0 :

```
 async def handle_present_proof_v2_0(self, message):
    state = message["state"]
    pres_ex_id = message["pres_ex_id"]
    self.log(f"Presentation: state = {state}, pres_ex_id = {pres_ex_id}")

    if state == "presentation-received":
        log_status("#27 Process the proof provided by X")
        log_status("#28 Check if proof is valid")
        proof = await self.admin_POST(
            f"/present-proof-2.0/records/{pres_ex_id}/verify-presentation"
        )
        self.log("Proof = ", proof["verified"])

        # Si la présentation est un schéma de diplôme (degree schema)
        # nous vérifions les valeurs reçues
        pres_req = message["by_format"]["pres_request"]["indy"]
        pres = message["by_format"]["pres"]["indy"]
        is_proof_of_education = (
            pres_req["name"] == "Proof of Education"
        )
        if is_proof_of_education:
            log_status("#28.1 Received proof of education, check claims")
            for (referent, attr_spec) in pres_req["requested_attributes"].items():
                self.log(
                    f"{attr_spec['name']}: "
                    f"{pres['requested_proof']['revealed_attrs'][referent]
['raw']}"
                )
            for id_spec in pres["identifiers"]:
                self.log(f"schema_id: {id_spec['schema_id']}")
                self.log(f"cred_def_id {id_spec['cred_def_id']}")
            # TODO placeholder for the next step
        else:
            self.log("#28.1 Received ", message["presentation_request"]["name"])
```

Bien évidemment, plus d'informations sur la méthode API "/present-proof-2.0/records/{pres_ex_id}/verify-presentation" peut être trouvé sur Swagger :



Envoi de credential

Pour commencer, il est nécessaire d'enregistrer un nouveau schéma ainsi que de définir les crédentiels. Le tutoriel nous indique donc de trouver ce code et de retirer les commentaires de tous les éléments en commentaires et d'ajouter le code pour publier le schéma et la définition. Le code final est le suivant :

```
acme_schema_name = "employee id schema"
acme_schema_attrs = ["employee_id", "name", "date", "position"]
await acme_agent.initialize(
    the_agent=agent,
    schema_name=acme_schema_name,
    schema_attrs=acme_schema_attrs,
)

with log_timer("Publish schema and cred def duration:"):
    # define schema
    version = format(
        "%d.%d.%d"
        % (
            random.randint(1, 101),
            random.randint(1, 101),
            random.randint(1, 101),
        )
    )
    # register schema and cred def
    (schema_id, cred_def_id) = await agent.register_schema_and_creddef(
```

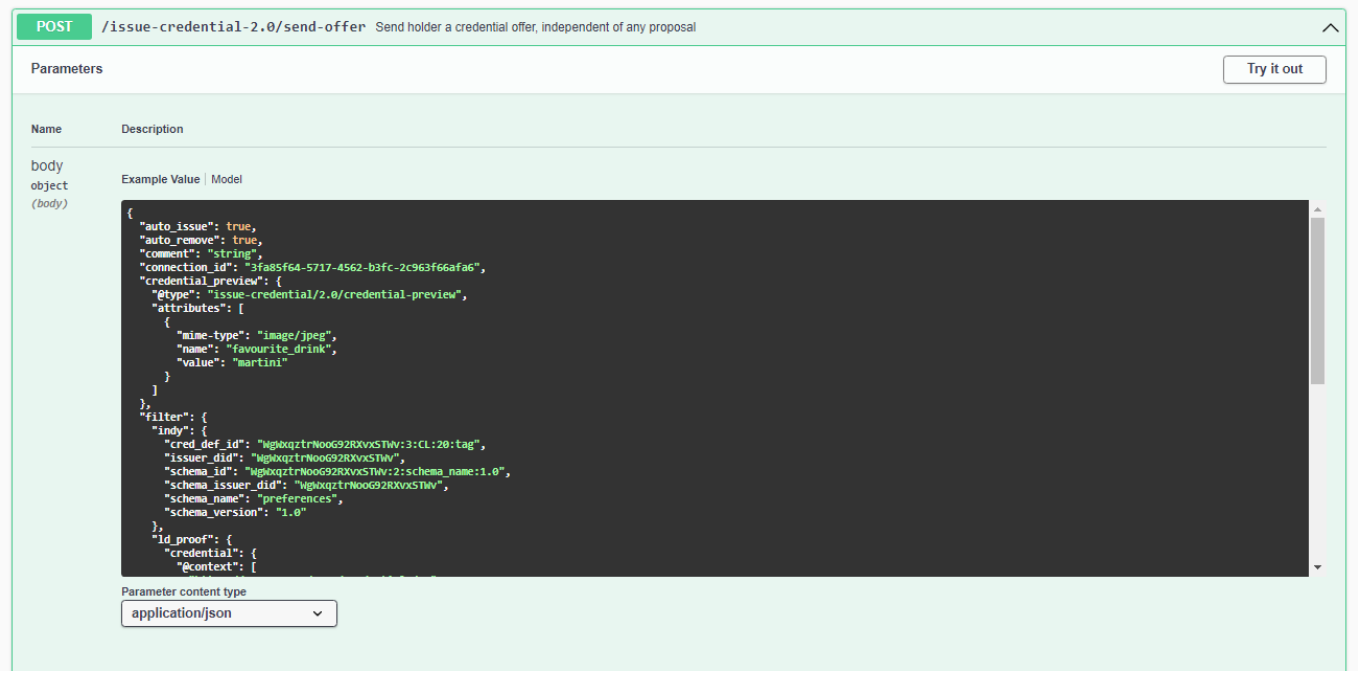
```
        "employee id schema",
        version,
        ["employee_id", "name", "date", "position"],
        support_revocation=False,
        revocation_registry_size=TAILS_FILE_COUNT,
    )
```

Bien évidemment en inspectant la méthode register_schema_and_creddef, nous voyons dans le code du fichier agent.py, que la méthode appelée s'occupe de contacter l'API via des POST et des GET pour enregistrer le schéma.

Ensuite, nous voulons compléter l'option "1", qui actionne l'envoi d'une proposition de crédentiels, avec le code suivant :

```
elif option == "1":
    log_status("#13 Issue credential offer to X")
    agent.cred_attrs[cred_def_id] = {
        "employee_id": "ACME0009",
        "name": "Alice Smith",
        "date": date.isoformat(date.today()),
        "position": "CEO"
    }
    cred_preview = {
        "@type": CRED_PREVIEW_TYPE,
        "attributes": [
            {"name": n, "value": v}
            for (n, v) in agent.cred_attrs[cred_def_id].items()
        ],
    }
    offer_request = {
        "connection_id": agent.connection_id,
        "comment": f"Offer on cred def id {cred_def_id}",
        "credential_preview": cred_preview,
        "filter": {"indy": {"cred_def_id": cred_def_id}},
    }
    await agent.admin_POST(
        "/issue-credential-2.0/send-offer", offer_request
    )
```

Comme dans les exemples précédents, il est bien évidemment possible d'avoir plus d'informations sur la méthode "/issue-credential-2.0/send-offer" via Swagger.



La dernière partie manquante est celle lorsque l'état de acme est en "request-received", c'est-à-dire qu'Alice a accepté la proposition de crédentiels, et que l'acme publie celui-ci. Le code est le suivant :

```
if state == "request-received":
    if not message.get("auto_issue"):
        await self.admin_POST(
            f"/issue-credential-2.0/records/{cred_ex_id}/issue",
            {"comment": f"Issuing credential, exchange {cred_ex_id}"},
        )
```

Test Complet du Scénario

Nous allons maintenant effectuer le scénario complet décrit dans la première partie du tutoriel.

Depuis le dossier "aries-cloudagent-python/demo", nous allons lancer deux shell. Dans le premier, nous lançons Faber avec le ledger de greenlight.

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-cloudagent-python/demo$
LEDGER_URL=http://dev.greenlight.bcovrin.vonx.io ./run_demo faber
...
Invitation Data:
{"@type": "https://didcomm.org/out-of-band/1.0/invitation", "@id": "6ab327f6-901d-
40e3-8363-afb8f3bbf263", "services": [{"id": "#inline", "type": "did-
communication", "recipientKeys":
["did:key:z6MkjxPogsuxkEnHjoDMzvVrb9D149Zjqe47SW2xKX82Ucnb"], "serviceEndpoint":
"http://192.168.65.3:8020"}], "label": "faber.agent", "handshake_protocols":
["https://didcomm.org/didexchange/1.0"]}
...
Waiting for connection...
```

Puis, nous lançons Alice sur le même ledger et copions l'invitation de Faber :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-cloudagent-python/demo$
LEDGER_URL=http://dev.greenlight.bcovrin.vonx.io ./run_demo alice
...
#9 Input faber.py invitation details
Invite details: {"@type": "https://didcomm.org/out-of-band/1.0/invitation", "@id":
"6ab327f6-901d-40e3-8363-afb
8f3bbf263", "services": [{"id": "#inline", "type": "did-communication",
"recipientKeys": ["did:key:z6MkjxPogsux
kEnHjoDMzvVrb9D149Zjqe47SW2xKX82Ucnb"], "serviceEndpoint":
"http://192.168.65.3:8020"}], "label": "faber.agent"
, "handshake_protocols": ["https://didcomm.org/didexchange/1.0"]}
Invitation response:
  {
    "my_did": "RU6MfBHkVvDTjRFqWXXvtD",
    "invitation_key": "6W8m6dfXQhHpdJNfKMY1k3f1EaHtRkokkV82VFA1ZQ1D",
    "request_id": "a8106f5c-8721-40a6-90fc-02dcd1ae5335",
    "their_label": "faber.agent",
    "invitation_msg_id": "6ab327f6-901d-40e3-8363-afb8f3bbf263",
    "invitation_mode": "once",
    "their_role": "inviter",
    "connection_protocol": "didexchange/1.0",
    "created_at": "2022-05-18T15:52:05.264295Z",
    "accept": "auto",
    "routing_state": "none",
    "state": "request",
    "updated_at": "2022-05-18T15:52:05.297282Z",
    "rfc23_state": "request-sent",
    "connection_id": "e24da72d-1d7b-4c4d-9bed-4a306262c4a7"
  }

Connect duration: 0.05s
Waiting for connection...
Alice      | Connected
Alice      | Check for endorser role ...
Connect duration: 0.15s
    (3) Send Message
    (4) Input New Invitation
    (X) Exit?
[3/4/X]
```

Depuis le shell de Faber, il est maintenant possible via "1" d'envoyer un crédentiel à Alice et ensuite nous pouvons quitter le shell de Faber avec "x" :

```
[1/2/3/4/T/X] 1

#13 Issue credential offer to X
Faber      | Credential: state = offer-sent, cred_ex_id = fe25d2a7-5af3-4552-91dc-
5cc2c62f9a48
```

```
Faber      | Credential: state = request-received, cred_ex_id = fe25d2a7-5af3-
4552-91dc-5cc2c62f9a48

#17 Issue credential to X
Faber      | Credential: state = credential-issued, cred_ex_id = fe25d2a7-5af3-
4552-91dc-5cc2c62f9a48
Faber      | Credential: state = done, cred_ex_id = fe25d2a7-5af3-4552-91dc-
5cc2c62f9a48
    (1) Issue Credential
    (2) Send Proof Request
    (2a) Send *Connectionless* Proof Request (requires a Mobile client)
    (3) Send Message
    (4) Create New Invitation
    (T) Toggle tracing on credential/proof exchange
    (X) Exit?
[1/2/3/4/T/X] x
Shutting down agent ...
Faber      |
Faber      | Shutting down
Faber      | Exited with return code 0
```

Du côté d'Alice, nous voyons l'élément suivant :

```
#18.1 Stored credential c63fa010-15ce-4075-9a2c-3408f6801cd9 in wallet
Credential details:
  {
    "referent": "c63fa010-15ce-4075-9a2c-3408f6801cd9",
    "attrs": {
      "date": "2018-05-28",
      "timestamp": "1652889232",
      "birthdate_dateint": "19980518",
      "name": "Alice Smith",
      "degree": "Maths"
    },
    "schema_id": "KcV47U9edQi2NbjkJuJzLF:2:degree schema:32.95.31",
    "cred_def_id": "KcV47U9edQi2NbjkJuJzLF:3:CL:131995:faber.agent.degree_schema",
    "rev_reg_id": null,
    "cred_rev_id": null
  }

Alice      | credential_id c63fa010-15ce-4075-9a2c-3408f6801cd9
Alice      | cred_def_id
KcV47U9edQi2NbjkJuJzLF:3:CL:131995:faber.agent.degree_schema
Alice      | schema_id KcV47U9edQi2NbjkJuJzLF:2:degree schema:32.95.31
Alice      | Credential: state = done, cred_ex_id = e2b16412-9246-477b-9dbe-
b5b6fe20fee0
    (3) Send Message
    (4) Input New Invitation
    (X) Exit?
[3/4/X]
```

Nous pouvons noter que le crédentiel réceptionné par Alice contient les attributs que l'ACME utilisé pour contrôler qu'elle est éligible.

Maintenant, nous pouvons lancer ACME dans un nouveau shell :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-cloudagent-python/demo$
LEDGER_URL=http://dev.greenlight.bcovrin.vonx.io ./run_demo acme
...
Invitation Data:
{"@type": "https://didcomm.org/out-of-band/1.0/invitation", "@id": "01ecb90a-fc22-
4ab6-b592-88990e813a32", "label": "acme.agent", "services": [{"id": "#inline",
"type": "did-communication", "recipientKeys":
["did:key:z6MkuWrwyayKhEiWozXeRbB1S9CAeh8pxFGzd1VZuiJgXsh8"], "serviceEndpoint":
"http://192.168.65.3:8040"}], "handshake_protocols":
["https://didcomm.org/didexchange/1.0"]}
...
Waiting for connection...
```

Comme pour Faber, il suffit de copier les informations d'invitation d'ACME après avoir appuyé sur "4" sur le shell d'Alice :

```
[3/4/X] 4

Input new invitation details
Invite details: {
   "@type":"https://didcomm.org/out-of-band/1.0/invitation",
   "@id":"01ecb90a-fc22-4ab6-b592-88990e813a32",
   "label":"acme.agent",
   "services":[
      {
         "id":"#inline",
         "type":"did-communication",
         "recipientKeys":[
            "did:key:z6MkuWrwyayKhEiWozXeRbB1S9CAeh8pxFGzd1VZuiJgXsh8"
         ],
         "serviceEndpoint":"http://192.168.65.3:8040"
      }
   ],
   "handshake_protocols":[
      "https://didcomm.org/didexchange/1.0"
   ]
}
Invitation response:
  {
    "my_did": "9gAEtfuEHffYua15entv5P",
    "invitation_key": "G4buPLitMhE3hVgwk2DAb3eAq7ryYN2dvzae5SLfceuk",
    "request_id": "74d735be-5506-46ae-a237-5ec0162ff29d",
    "their_label": "acme.agent",
    "invitation_msg_id": "01ecb90a-fc22-4ab6-b592-88990e813a32",
    "invitation_mode": "once",
```

```
    "their_role": "inviter",
    "connection_protocol": "didexchange/1.0",
    "created_at": "2022-05-18T16:06:44.156126Z",
    "accept": "auto",
    "routing_state": "none",
    "state": "request",
    "updated_at": "2022-05-18T16:06:44.188733Z",
    "rfc23_state": "request-sent",
    "connection_id": "aff000fb-a14c-4b18-8d75-9786fcc5f03f"
  }

Connect duration: 0.05s
Waiting for connection...
Alice      | Connected
Alice      | Check for endorser role ...
Connect duration: 0.14s
    (3) Send Message
    (4) Input New Invitation
    (X) Exit?
[3/4/X]
```

Maintenant depuis le shell d'ACME, grâce aux options que nous avons développées, nous pouvons utiliser l'option "2" pour demander une proof :

```
[1/2/3/X]2

#20 Request proof of degree from alice
Acme       | Presentation: state = request-sent, pres_ex_id = c40f6d5d-81fc-426b-
9c50-7529c8c19ce1
Acme       | Presentation: state = presentation-received, pres_ex_id = c40f6d5d-
81fc-426b-9c50-7529c8c19ce1

#27 Process the proof provided by X

#28 Check if proof is valid
Acme       | Presentation: state = done, pres_ex_id = c40f6d5d-81fc-426b-9c50-
7529c8c19ce1
Acme       | Proof =  true

#28.1 Received proof of education, check claims
Acme       | name: Alice Smith
Acme       | date: 2018-05-28
Acme       | degree: Maths
Acme       | schema_id: KcV47U9edQi2NbjkJuJzLF:2:degree schema:32.95.31
Acme       | cred_def_id
KcV47U9edQi2NbjkJuJzLF:3:CL:131995:faber.agent.degree_schema
    (1) Issue Credential
    (2) Send Proof Request
    (3) Send Message
    (X) Exit?
[1/2/3/X]
```

Du côté d'Alice, nous pouvons voir que nous avons reçu la requête de proof et l'envoi automatique de la proof :

```
Alice      | Presentation: state = request-received, pres_ex_id = 2806ed58-20cf-
4971-86d9-35b34fa5c20f

#24 Query for credentials in the wallet that satisfy the proof request

#25 Generate the proof

#26 Send the proof to X: {"indy": {"requested_predicates": {},
"requested_attributes": {"0_name_uuid": {"cred_id": "c63fa010-15ce-4075-9a2c-
3408f6801cd9", "revealed": true}, "0_date_uuid": {"cred_id": "c63fa010-15ce-4075-
9a2c-3408f6801cd9", "revealed": true}, "0_degree_uuid": {"cred_id": "c63fa010-
15ce-4075-9a2c-3408f6801cd9", "revealed": true}}, "self_attested_attributes": {}}}
Alice      | Presentation: state = presentation-sent, pres_ex_id = 2806ed58-20cf-
4971-86d9-35b34fa5c20f
Alice      | Presentation: state = done, pres_ex_id = 2806ed58-20cf-4971-86d9-
35b34fa5c20f
Alice      | Credential: state = offer-received, cred_ex_id = d9b2d2e3-4719-4ba9-
a120-186b36805d1d
```

Puis, nous pouvons envoyer le crédentiel à Alice avec l'option "1" sur ACME :

```
[1/2/3/X]1

#13 Issue credential offer to X
Acme       | Credential: state = offer-sent, cred_ex_id = 5d67a1be-21f6-4a58-8776-
691de94901a6
Acme       | Credential: state = request-received, cred_ex_id = 5d67a1be-21f6-
4a58-8776-691de94901a6
Acme       | Credential: state = credential-issued, cred_ex_id = 5d67a1be-21f6-
4a58-8776-691de94901a6
Acme       | Credential: state = done, cred_ex_id = 5d67a1be-21f6-4a58-8776-
691de94901a6
    (1) Issue Credential
    (2) Send Proof Request
    (3) Send Message
    (X) Exit?
[1/2/3/X]
```

Finalement, du côté d'Alice, nous voyons le crédentiel reçu qui comprend les mêmes attributs que ceux que nous avons défini dans le code :

```
#15 After receiving credential offer, send credential request
Alice      | Credential: state = request-sent, cred_ex_id = d9b2d2e3-4719-4ba9-
a120-186b36805d1d
```

```
Alice      | Credential: state = credential-received, cred_ex_id = d9b2d2e3-4719-
4ba9-a120-186b36805d1d

#18.1 Stored credential 8fba1ae1-3dc3-4278-a5fa-d604c3ec49bc in wallet
Credential details:
  {
    "referent": "8fba1ae1-3dc3-4278-a5fa-d604c3ec49bc",
    "attrs": {
      "employee_id": "ACME0009",
      "date": "2022-05-18",
      "position": "CEO",
      "name": "Alice Smith"
    },
    "schema_id": "VnevPopteN541RWWBnk7tB:2:employee id schema:87.14.34",
    "cred_def_id":
"VnevPopteN541RWWBnk7tB:3:CL:132005:acme.agent.employee_id_schema",
    "rev_reg_id": null,
    "cred_rev_id": null
  }

Alice      | credential_id 8fba1ae1-3dc3-4278-a5fa-d604c3ec49bc
Alice      | cred_def_id
VnevPopteN541RWWBnk7tB:3:CL:132005:acme.agent.employee_id_schema
Alice      | schema_id VnevPopteN541RWWBnk7tB:2:employee id schema:87.14.34
Alice      | Credential: state = done, cred_ex_id = d9b2d2e3-4719-4ba9-a120-
186b36805d1d
    (3) Send Message
    (4) Input New Invitation
    (X) Exit?
[3/4/X]
```

HyperLedger Aries - Exploration de la Framework Go

Ce document montre la reproduction du lab suivant sur une machine wsl2 Ubuntu sur Windows 11.

Prérequis

Pour effectuer ce Lab, il est nécessaire d'avoir :

- Docker
- Docker Compose
- Go 1.16 ou plus
- Make
- Bash
- Node.js

Avant-Propos sur Aries Go

L'architecture de la Framework en Go est la même que celle en ACA-Py, cela signifie que nous avons une interface HTTP entre un contrôleur et la Framework. Ainsi, un contrôleur écrit pour ACA-Py devrait fonctionner avec une instance de la Framework en Go.

Ce qui est change avec cette Framework, c'est le fait que celle-ci est complètement écrite en Go ainsi il n'y a aucune dépendance "non-golang". Mais cela implique aussi que la Framework n'a pas l'Indy-sdk et ainsi ne supporte pas la connexion aux ledgers Indy ou l'utilisation du modèle d'échange de crédentiels d'Indy.

L'équipe d'Aries Go construit cependant un support vers d'autres ledgers et d'autres modèles d'échanges.

Étapes de réalisation

Mise en place de l'environnement du lab

Comme nous l'indique le lab, il est important d'avoir certains prérequis installer pour pouvoir utiliser la Framework. La liste de ceux-ci est donnée plus haut. Si plus d'informations sont nécessaires sur ces prérequis, les informations sont trouvables ici. C'est également sur ce lien que nous trouvons les informations sur l'utilisation de la commande "Make" ainsi que la génération de matériels Cryptos.

La première étape de réalisation va simplement être de cloner le repository sur notre machine.

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython$ git clone
https://github.com/hyperledger/aries-Framework-go
...
valonrexhepi@WINDOWSDESKTOP:~/AriesPython$ cd aries-Framework-go
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$
```

Une fois le repository disponible localement, nous pouvons lancer la démo en utilisant la commande "make" depuis le root du repository :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$ make run-openapi-
demo
...
5.5s
[+] Running 6/6
 ⠿ Container alice.openapi.demo.com        Started
1.0s
 ⠿ Container carl.router.openapi.demo.com  Started
2.1s
 ⠿ Container bob.openapi.demo.com          Started
2.1s
 ⠿ Container dave.router.openapi.demo.com  Started
1.8s
 ⠿ Container carl.openapi.demo.com         Started
1.4s
 ⠿ Container dave.openapi.demo.com         Started
1.8s
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$
```

De nombreux éléments viennent d'être téléchargés et lancés. Une fois que tous les containers sont opérationnels, comme ci-dessus, il est maintenant possible de lancer des interfaces Swagger pour l'OpenAPI.

Étapes pour le DIDExchange

Maintenant que la démo est lancée, nous allons nous rendre sur l'interface OpenAPI d'Alice et de Bob pour effectuer un échange de DID entre les deux.

Nous allons commencer par lancer une requête pour créer une invitation depuis l'agent d'Alice :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$ curl -k -X 'POST'
'https://localhost:8082/connections/create-invitation' -H 'accept:
application/json' -d '' | python3 -mjson.tool
{
    "invitation": {
        "serviceEndpoint": "https://alice.aries.example.com:8081",
        "recipientKeys": [
            "did:key:z6LStiQYe9ihKipezr8cZ4MRKWdTFjfBpgmgDuXWJRdkNkZm"
        ],
        "@id": "c1cff2d2-05f5-40b0-90ac-d0ccda294c76",
        "label": "alice-agent",
        "@type": "https://didcomm.org/didexchange/1.0/invitation"
    },
    "alias": "",
    "invitation_url": ""
}
```

Nous pouvons maintenant copier l'invitation sur l'agent de Bob sur la méthode pour sauvegarder l'invitation :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$ curl -k -X 'POST'
'https://localhost:9082/connections/receive-invitation' -H 'accept:
application/json' -H 'Content-Type: application/json' \
  -d '{
    "serviceEndpoint":"https://alice.aries.example.com:8081",
    "recipientKeys":[
       "did:key:z6LStiQYe9ihKipezr8cZ4MRKWdTFjfBpgmgDuXWJRdkNkZm"
    ],
    "@id":"c1cff2d2-05f5-40b0-90ac-d0ccda294c76",
    "label":"alice-agent",
    "@type":"https://didcomm.org/didexchange/1.0/invitation"
 }'
{"state":"","created_at":"0001-01-01T00:00:00Z","updated_at":"0001-01-
01T00:00:00Z","connection_id":"dd6b3a8b-1855-4bf5-b75e-
3014684dd29d","request_id":"","my_did":""}
```

Avec le connection ID ci-dessus, nous pouvons fetch la connexion pour voir l'état actuel en "invited" depuis l'agent de Bob :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$ curl -k -X 'GET'
'https://localhost:9082/connections/dd6b3a8b-1855-4bf5-b75e-3014684dd29d' -H
'accept: application/json' | python3 -mjson.tool
{
    "result": {
        "ConnectionID": "dd6b3a8b-1855-4bf5-b75e-3014684dd29d",
        "State": "invited",
        "ThreadID": "c1cff2d2-05f5-40b0-90ac-d0ccda294c76",
        "ParentThreadID": "",
        "TheirLabel": "alice-agent",
        "TheirDID": "",
        "MyDID": "",
        "ServiceEndPoint": {
            "uri": "https://alice.aries.example.com:8081"
        },
        "RecipientKeys": [
            "did:key:z6LStiQYe9ihKipezr8cZ4MRKWdTFjfBpgmgDuXWJRdkNkZm"
        ],
        "InvitationID": "c1cff2d2-05f5-40b0-90ac-d0ccda294c76",
        "InvitationDID": "",
        "Implicit": false,
        "Namespace": "my",
        "DIDCommVersion": "v1",
        "PeerDIDInitialState": ""
    }
}
```

Si nous récupérons toutes les connexions disponibles, nous pouvons voir qu'il y en a une dont l'état est en "invited" :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$ curl -k -X 'GET'
'https://localhost:9082/connections' -H 'accept: application/json' | python3 -
mjson.tool
{
    "results": [
        {
            "ConnectionID": "dd6b3a8b-1855-4bf5-b75e-3014684dd29d",
            "State": "invited",
            "ThreadID": "c1cff2d2-05f5-40b0-90ac-d0ccda294c76",
            "ParentThreadID": "",
            "TheirLabel": "alice-agent",
            "TheirDID": "",
            "MyDID": "",
            "ServiceEndPoint": {
                "uri": "https://alice.aries.example.com:8081"
            },
            "RecipientKeys": [
                "did:key:z6LStiQYe9ihKipezr8cZ4MRKWdTFjfBpgmgDuXWJRdkNkZm"
            ],
            "InvitationID": "c1cff2d2-05f5-40b0-90ac-d0ccda294c76",
            "InvitationDID": "",
            "Implicit": false,
            "Namespace": "my",
            "DIDCommVersion": "v1",
            "PeerDIDInitialState": ""
        }
    ]
}
```

Depuis l'agent de Bob, nous pouvons maintenant accepter l'invitation d'Alice en passant l'ID de connexion, l'état de la connexion passe ensuite en "requested" :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$ curl -k -X 'POST'
'https://localhost:9082/connections/dd6b3a8b-1855-4bf5-b75e-3014684dd29d/accept-
invitation' -H 'accept: application/json' -d '' | python3 -mjson.tool
{
   "created_at":"0001-01-01T00:00:00Z",
   "updated_at":"0001-01-01T00:00:00Z",
   "connection_id":"dd6b3a8b-1855-4bf5-b75e-3014684dd29d"
}
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$ curl -k -X 'GET'
'https://localhost:9082/connections' -H 'accept: application/json' | python3 -
mjson.tool
{
    "results": [
        {
            "ConnectionID": "dd6b3a8b-1855-4bf5-b75e-3014684dd29d",
            "State": "requested",
            "ThreadID": "c1cff2d2-05f5-40b0-90ac-d0ccda294c76",
            "ParentThreadID": "",
            "TheirLabel": "alice-agent",
```

```
            "TheirDID": "",
            "MyDID": "did:peer:1zQmaSoD3tfSHE7VYsxmJueEedSm4UnHaPSFk4ySXu8YUW3t",
            "ServiceEndPoint": {
                "uri": "https://alice.aries.example.com:8081"
            },
            "RecipientKeys": [
                "did:key:z6LStiQYe9ihKipezr8cZ4MRKWdTFjfBpgmgDuXWJRdkNkZm"
            ],
            "InvitationID": "c1cff2d2-05f5-40b0-90ac-d0ccda294c76",
            "InvitationDID": "",
            "Implicit": false,
            "Namespace": "my",
            "DIDCommVersion": "v1",
            "PeerDIDInitialState": ""
        }
    ]
}
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$ curl -k -X 'GET'
'https://localhost:8082/connections' -H 'accept: application/json' | python3 -
mjson.tool
{
    "results": [
        {
            "ConnectionID": "e70f5039-42b7-42a5-a936-baff983c2691",
            "State": "requested",
            "ThreadID": "c1cff2d2-05f5-40b0-90ac-d0ccda294c76",
            "ParentThreadID": "",
            "TheirLabel": "bob-agent",
            "TheirDID":
"did:peer:1zQmaSoD3tfSHE7VYsxmJueEedSm4UnHaPSFk4ySXu8YUW3t",
            "MyDID": "",
            "ServiceEndPoint": {
                "uri": ""
            },
            "RecipientKeys": null,
            "InvitationID": "c1cff2d2-05f5-40b0-90ac-d0ccda294c76",
            "InvitationDID": "",
            "Implicit": false,
            "Namespace": "their",
            "DIDCommVersion": "v1",
            "PeerDIDInitialState": ""
        }
    ]
}
```

Depuis l'agent d'Alice, il est maintenant possible d'accepter la requête de Bob avec l'ID connexion que nous venons de récupérer :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$ curl -k -X 'POST'
'https://localhost:8082/connections/e70f5039-42b7-42a5-a936-baff983c2691/accept-
request' -H 'accept: application/json' -d '' | python3 -mjson.tool
```

```
{
   "their_did":"",
   "request_id":"",
   "connection_id":"e70f5039-42b7-42a5-a936-baff983c2691",
   "updated_at":"0001-01-01T00:00:00Z",
   "created_at":"0001-01-01T00:00:00Z",
   "state":""
}
```


Si nous effectuons une récupération sur la connexion avec l'ID de connexion ci-dessus, l'état est maintenant "completed" :


```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$ curl -k -X 'GET'
'https://localhost:8082/connections/e70f5039-42b7-42a5-a936-baff983c2691' -H
'accept: application/json' | python3 -mjson.tool
{
    "result": {
        "ConnectionID": "e70f5039-42b7-42a5-a936-baff983c2691",
        "State": "completed",
        "ThreadID": "c1cff2d2-05f5-40b0-90ac-d0ccda294c76",
        "ParentThreadID": "",
        "TheirLabel": "bob-agent",
        "TheirDID": "did:peer:1zQmaSoD3tfSHE7VYsxmJueEedSm4UnHaPSFk4ySXu8YUW3t",
        "MyDID": "did:peer:1zQmTnNPm8kfT9DUrC6wEqmZNTok72SU937E2E9UNAj4LgTf",
        "ServiceEndPoint": {
            "uri": "",
            "accept": [
                "didcomm/v2"
            ]
        },
        "RecipientKeys": null,
        "InvitationID": "c1cff2d2-05f5-40b0-90ac-d0ccda294c76",
        "InvitationDID": "",
        "Implicit": false,
        "Namespace": "their",
        "DIDCommVersion": "v1",
        "PeerDIDInitialState": ""
    }
}
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$ curl -k -X 'GET'
'https://localhost:9082/connections' -H 'accept: application/json' | python3 -
mjson.tool
{
    "results": [
        {
            "ConnectionID": "dd6b3a8b-1855-4bf5-b75e-3014684dd29d",
            "State": "completed",
            "ThreadID": "c1cff2d2-05f5-40b0-90ac-d0ccda294c76",
            "ParentThreadID": "",
            "TheirLabel": "alice-agent",
            "TheirDID":
```

```
"did:peer:1zQmTnNPm8kfT9DUrC6wEqmZNTok72SU937E2E9UNAj4LgTf",
            "MyDID": "did:peer:1zQmaSoD3tfSHE7VYsxmJueEedSm4UnHaPSFk4ySXu8YUW3t",
            "ServiceEndPoint": {
                "uri": "https://alice.aries.example.com:8081"
            },
            "RecipientKeys": [
                "did:key:z6LStiQYe9ihKipezr8cZ4MRKWdTFjfBpgmgDuXWJRdkNkZm"
            ],
            "InvitationID": "c1cff2d2-05f5-40b0-90ac-d0ccda294c76",
            "InvitationDID": "",
            "Implicit": false,
            "Namespace": "my",
            "DIDCommVersion": "v1",
            "PeerDIDInitialState": ""
        }
    ]
}
```

Étapes pour la gestion de messages

Si vous n'avez pas arrêté le docker depuis l'exemple précédent, vous pouvez passer cette étape. Nous allons recréer une connexion entre Alice et Bob. Tout d'abord, nous créons une invitation sur l'agent d'Alice :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$ curl -k -X 'POST'
'https://localhost:8082/connections/create-invitation' -H 'accept:
application/json' -d '' | python3 -mjson.tool
{
    "invitation": {
        "serviceEndpoint": "https://alice.aries.example.com:8081",
        "recipientKeys": [
            "did:key:z6LSkDBcMnuVyPzg5FSZGQSVNVkYNVi51osUJE76Eh7Uwxvi"
        ],
        "@id": "534aa9b3-520b-44dc-9f4b-9cd7a7c1c7ee",
        "label": "alice-agent",
        "@type": "https://didcomm.org/didexchange/1.0/invitation"
    },
    "alias": "",
    "invitation_url": ""
}
```

Puis nous recevons l'invitation depuis Bob :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$ curl -k -X 'POST'
'https://localhost:9082/connections/receive-invitation' -H 'accept:
application/json' -H 'Content-Type: application/json' \
   -d '{
    "serviceEndpoint":"https://alice.aries.example.com:8081",
    "recipientKeys":[
       "did:key:z6LSkDBcMnuVyPzg5FSZGQSVNVkYNVi51osUJE76Eh7Uwxvi"
```

```
    ],
    "@id":"534aa9b3-520b-44dc-9f4b-9cd7a7c1c7ee",
    "label":"alice-agent",
    "@type":"https://didcomm.org/didexchange/1.0/invitation"
 }'
{
   "state":"",
   "created_at":"0001-01-01T00:00:00Z",
   "updated_at":"0001-01-01T00:00:00Z",
   "connection_id":"922ae76d-4b73-4d39-9f3b-238ef89c0361",
   "request_id":"",
   "my_did":""
}
```

Nous acceptons ensuite l'invitation :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$ curl -k -X 'POST'
'https://localhost:9082/connections/922ae76d-4b73-4d39-9f3b-238ef89c0361/accept-
invitation' -H 'accept: application/json' -d '' | python3 -mjson.tool
{
    "created_at": "0001-01-01T00:00:00Z",
    "updated_at": "0001-01-01T00:00:00Z",
    "connection_id": "922ae76d-4b73-4d39-9f3b-238ef89c0361"
}
```

Nous affichons ensuite les connexions d'Alice :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$ curl -k -X 'GET'
'https://localhost:8082/connections' -H 'accept: application/json' | python3 -
mjson.tool
{
    "results": [
        {
            "ConnectionID": "a9e6d389-a1e5-45ee-ad7f-11669e3ea9b6",
            "State": "requested",
            "ThreadID": "534aa9b3-520b-44dc-9f4b-9cd7a7c1c7ee",
            "ParentThreadID": "",
            "TheirLabel": "bob-agent",
            "TheirDID":
"did:peer:1zQmeMKxbHYE1HxTQthzhvJjpMvdtEb7u7svSfooZo6Ckffs",
            "MyDID": "",
            "ServiceEndPoint": {
                "uri": ""
            },
            "RecipientKeys": null,
            "InvitationID": "534aa9b3-520b-44dc-9f4b-9cd7a7c1c7ee",
            "InvitationDID": "",
            "Implicit": false,
            "Namespace": "their",
```

```
            "DIDCommVersion": "v1",
            "PeerDIDInitialState": ""
        }
    ]
}
```

Pour pouvoir ensuite accepter la requête :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$ curl -k -X 'POST'
'https://localhost:8082/connections/a9e6d389-a1e5-45ee-ad7f-11669e3ea9b6/accept-
request' -H 'accept: application/json' -d '' | python3 -mjson.tool
{
    "their_did": "",
    "request_id": "",
    "connection_id": "a9e6d389-a1e5-45ee-ad7f-11669e3ea9b6",
    "updated_at": "0001-01-01T00:00:00Z",
    "created_at": "0001-01-01T00:00:00Z",
    "state": ""
}
```

La connexion est complète chez Alice et Bob :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$ curl -k -X 'GET'
'https://localhost:8082/connections' -H 'accept: application/json' | python3 -
mjson.tool
{
    "results": [
        {
            "ConnectionID": "a9e6d389-a1e5-45ee-ad7f-11669e3ea9b6",
            "State": "completed",
            "ThreadID": "534aa9b3-520b-44dc-9f4b-9cd7a7c1c7ee",
            "ParentThreadID": "",
            "TheirLabel": "bob-agent",
            "TheirDID":
"did:peer:1zQmeMKxbHYE1HxTQthzhvJjpMvdtEb7u7svSfooZo6Ckffs",
            "MyDID": "did:peer:1zQmYgcD5Dm6fhou6jamhLYVqSyyFsc1DcVGizUCzmWDQNv3",
            "ServiceEndPoint": {
                "uri": "",
                "accept": [
                    "didcomm/v2"
                ]
            },
            "RecipientKeys": null,
            "InvitationID": "534aa9b3-520b-44dc-9f4b-9cd7a7c1c7ee",
            "InvitationDID": "",
            "Implicit": false,
            "Namespace": "their",
            "DIDCommVersion": "v1",
            "PeerDIDInitialState": ""
```

```
        }
    ]
}
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$ curl -k -X 'GET'
'https://localhost:9082/connections' -H 'accept: application/json' | python3 -
mjson.tool
{
    "results": [
        {
            "ConnectionID": "922ae76d-4b73-4d39-9f3b-238ef89c0361",
            "State": "completed",
            "ThreadID": "534aa9b3-520b-44dc-9f4b-9cd7a7c1c7ee",
            "ParentThreadID": "",
            "TheirLabel": "alice-agent",
            "TheirDID":
"did:peer:1zQmYgcD5Dm6fhou6jamhLYVqSyyFsc1DcVGizUCzmWDQNv3",
            "MyDID": "did:peer:1zQmeMKxbHYE1HxTQthzhvJjpMvdtEb7u7svSfooZo6Ckffs",
            "ServiceEndPoint": {
                "uri": "https://alice.aries.example.com:8081"
            },
            "RecipientKeys": [
                "did:key:z6LSkDBcMnuVyPzg5FSZGQSVNVkYNVi51osUJE76Eh7Uwxvi"
            ],
            "InvitationID": "534aa9b3-520b-44dc-9f4b-9cd7a7c1c7ee",
            "InvitationDID": "",
            "Implicit": false,
            "Namespace": "my",
            "DIDCommVersion": "v1",
            "PeerDIDInitialState": ""
        }
    ]
}
```

Depuis Alice, nous allons maintenant enregistrer un message "generic-invite" pour le type "https://didcomm.org/generic/1.0/message" via la méthode "/message/register-service" :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$ curl -k -X 'POST' \
   'https://localhost:8082/message/register-service' \
   -H 'accept: application/json' \
   -H 'Content-Type: application/json' \
   -d '{
   "name": "generic-invite",
   "purpose": [
     "meeting","appointment","event"
   ],
   "type": "https://didcomm.org/generic/1.0/message"
 }'
```

Nous pouvons vérifier que le service a bien été enregistré via la méthode "/message/services" :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$ curl -k -X 'GET'
'https://localhost:8082/message/services' -H 'accept: application/json'
{
    "names":["generic-invite"]
}
```

Il est possible d'envoyer un message depuis Bob à Alice via la méthode "/message/send/" en précisant le connection_ID de la connexion avec Alice (il est aussi possible de remplacer le connection_ID par their_did et préciser le did d'Alice) :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$ curl -k -X 'POST' \
   'https://localhost:9082/message/send' \
   -H 'accept: application/json' \
   -H 'Content-Type: application/json' \
   -d '{
    "connection_ID":"922ae76d-4b73-4d39-9f3b-238ef89c0361",
    "message_body":{
       "@id":"2d071d1c-a47d-40e5-a0b7-f8234231ba9e",
       "@type":"https://didcomm.org/generic/1.0/message",
       "~purpose":[
          "meeting"
       ],
       "message":"Hey, meet me today at 4PM",
       "sender":"Bob"
    }
 }' | python3 -mjson.tool
{}
```

Il n'y a malheureusement pas de fonctionnalité pour voir les mails depuis Swagger alors il faut se rendre sur le webhook d'Alice pour voir le message reçu :

```
{
   "id":"6c4c22b2-781f-4097-88da-4ce5fdf65931",
   "topic":"didexchange_actions",
   "message":{
      "ProtocolName":"didexchange",
      "Message":{
         "@id":"534aa9b3-520b-44dc-9f4b-9cd7a7c1c7ee",
         "@type":"https://didcomm.org/didexchange/1.0/request",
         "did":"did:peer:1zQmeMKxbHYE1HxTQthzhvJjpMvdtEb7u7svSfooZo6Ckffs",
         "did_doc~attach":{
            "data":{

"base64":"eyJAY29udGV4dCI6WyJodHRwczovL3d3dy53My5vcmcvbnMvZGlkL3YxIl0sImlkIjoiZGlk
OnBlZXI6MXpRbWVNS3hiSFlFMUh4VFF0aHpodkpqcE12ZHRFYjd1N3N2U2Zvb1pvNkNrZmZzIiwidmVyaW
ZpY2F0aW9uTWV0aG9kIjpbeyJjb250cm9sbGVyIjoiIiwiaWQiOiIja2V5LTEiLCJwdWJsaWNLZXlCYXNl
NTgiOiIzY20xbVZOYVJVeWRuZ0RzVHU3TWtvRUhEZVI1MnlCdHhxRURhNjZ5Z1FHOCIsInR5cGUiOiJFZD
I1NTE5VmVyaWZpY2F0aW9uS2V5MjAxOCJ9LHsiY29udHJvbGxlciI6IiIsImlkIjoiI2tleS0yIiwicHVi
```

```
bGljS2V5QmFzZTU4IjoiNjk2V21ZakNGQlN2aGZIOEQySjZmRGlnWlF5VkVSUmtIeHhQRjlnbmFDVngiLC
J0eXBlIjoiWDI1NTE5S2V5QWdyZWVtZW50S2V5MjAxOSJ9XSwic2VydmljZSI6W3siaWQiOiI0YzNhZTlk
Mi1hMzI5LTQwZWMtYmM5MC03YTljMDMyOTdiNWIiLCJwcmlvcml0eSI6MCwicmVjaXBpZW50S2V5cyI6Wy
JkaWQ6cGVlcjoxelFtZU1LeGJIWUUxSHhUUXRoemh2SmpwTXZkdEViN3U3c3ZTZm9vWm82Q2tmZnMja2V5
LTIiXSwic2VydmljZUVuZHBvaW50Ijp7InVyaSI6Imh0dHBzOi8vYm9iLmFyaWVzLmV4YW1wbGUuY29tOj
kwODEiLCJhY2NlcHQiOlsiZGlkY29tbS92MiJdfSwidHlwZSI6IkRJRENvbW1NZXNzYWdpbmcifV0sImF1
dGhlbnRpY2F0aW9uIjpbIiNrZXktMSJdLCJhc3NlcnRpb25NZXRob2QiOlsiI2tleS0xIl0sImtleUFncm
VlbWVudCI6WyIja2V5LTIiXSwiY3JlYXRlZCI6IjIwMjItMDUtMjNUMTc6NTU6MDYuMzMzMjI4MTVaIiwi
dXBkYXRlZCI6IjIwMjItMDUtMjNUMTc6NTU6MDYuMzMzMjI4MTVaIn0="
            },
            "lastmod_time":"0001-01-01T00:00:00Z",
            "mime-type":"application/json"
         },
         "label":"bob-agent",
         "~thread":{
            "pthid":"534aa9b3-520b-44dc-9f4b-9cd7a7c1c7ee"
         }
      },
      "Properties":{
         "connectionID":"a9e6d389-a1e5-45ee-ad7f-11669e3ea9b6",
         "invitationID":"534aa9b3-520b-44dc-9f4b-9cd7a7c1c7ee"
      }
   }
}
```

Si nous décodons le message en base64, nous obtenons ceci :

```
{
   "@context":[
      "https://www.w3.org/ns/did/v1"
   ],
   "id":"did:peer:1zQmeMKxbHYE1HxTQthzhvJjpMvdtEb7u7svSfooZo6Ckffs",
   "verificationMethod":[
      {
         "controller":"",
         "id":"#key-1",
         "publicKeyBase58":"3cm1mVNaRUydngDsTu7MkoEHDeR52yBtxqEDa66ygQG8",
         "type":"Ed25519VerificationKey2018"
      },
      {
         "controller":"",
         "id":"#key-2",
         "publicKeyBase58":"696WmYjCFBSvhfH8D2J6fDigZQyVERRkHxxPF9gnaCVx",
         "type":"X25519KeyAgreementKey2019"
      }
   ],
   "service":[
      {
         "id":"4c3ae9d2-a329-40ec-bc90-7a9c03297b5b",
         "priority":0,
         "recipientKeys":[
            "did:peer:1zQmeMKxbHYE1HxTQthzhvJjpMvdtEb7u7svSfooZo6Ckffs#key-2"
```

```
         ],
         "serviceEndpoint":{
            "uri":"https://bob.aries.example.com:9081",
            "accept":[
               "didcomm/v2"
            ]
         },
         "type":"DIDCommMessaging"
      }
   ],
   "authentication":[
      "#key-1"
   ],
   "assertionMethod":[
      "#key-1"
   ],
   "keyAgreement":[
      "#key-2"
   ],
   "created":"2022-05-23T17:55:06.33322815Z",
   "updated":"2022-05-23T17:55:06.33322815Z"
}
```

Il est ensuite possible "d'oublier" le service enregistré avec la méthode "/message/unregister-service" :

```
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$ curl -k -X 'POST' \
   'https://localhost:8082/message/unregister-service' \
   -H 'accept: application/json' \
   -H 'Content-Type: application/json' \
   -d '{
   "name": "generic-invite"
 }'
valonrexhepi@WINDOWSDESKTOP:~/AriesPython/aries-Framework-go$ curl -k -X 'GET'
'https://localhost:8082/message/services' -H 'accept: application/json'
{
    "names":[]
}
```

Étapes pour la gestion de messages via HTTP over DIDComm

C'est exactement le même processus que ci-dessus, mais nous utilisons la méthode "/http-over-didcomm/register" pour enregistrer des gestionnaires de messages de type "http-over-didcomm".

Créer un DID en utilisant vdr

Pour créer un DID, nous pouvons utiliser la méthode "/vdr/did/create", par exemple via l'agent d'Alice :

```
valonrexhepi@WINDOWSDESKTOP:~$ curl -k -X 'POST' \
   'https://localhost:8082/vdr/did/create' \
```

```
    -H 'accept: application/json' \
    -d '{
     "method":"peer",
     "did":{
        "id":"did:example:1zQmanz2svbjxcYd4J3CtP6Jg6kw4nQpnZQioscz4oKhtLHk",
        "@context":[
           "https://w3id.org/did/v1"
        ],
        "verificationMethod":[
           {
              "controller":"did:example:123",
              "id":"e2cbb249-8c25-4e6e-8b92-b1ceee211c8c",
              "publicKeyBase58":"7qf5xCRSGP3NW6PAUonYLmq1LCz6Ux5ynek9nbzGgCnP",
              "type":"Ed25519VerificationKey2018"
           }
        ]
     },
     "opts":{
        "store":true
     }
  }' | python3 -mjson.tool
{
    "did": {
        "@context": [
            "https://w3id.org/did/v1"
        ],
        "id": "did:example:1zQmanz2svbjxcYd4J3CtP6Jg6kw4nQpnZQioscz4oKhtLHk",
        "verificationMethod": [
            {
                "controller": "did:example:123",
                "id": "e2cbb249-8c25-4e6e-8b92-b1ceee211c8c",
                "publicKeyBase58": "7qf5xCRSGP3NW6PAUonYLmq1LCz6Ux5ynek9nbzGgCnP",
                "type": "Ed25519VerificationKey2018"
            }
        ]
    }
}
```

Créer une connexion DID à travers le protocole out-of-band

Il y a plusieurs étapes pour créer une connexion à travers le protocole out-of-band. Reprenons l'exemple d'Alice et Bob. Nous allons donc tout d'abord créer une invitation via Alice :

```
valonrexhepi@WINDOWSDESKTOP:~$ curl -k -X POST
"https://localhost:8082/outofband/create-invitation" -H  "accept:
application/json" -H  "Content-Type: application/json" -d "{  \"label\":
\"Alice\"}" | python3 -mjson.tool
{
   "invitation":{
      "@id":"46226bf4-5ed3-49d5-a56b-868baafe26b9",
      "@type":"https://didcomm.org/out-of-band/1.0/invitation",
```

```
        "label":"Alice",
        "services":[
           {
              "id":"26fb640e-749f-4277-9ad4-fe1f90686439",
              "type":"DIDCommMessaging",
              "recipientKeys":[
                 "did:key:z6LSr3Ti3jghy29UDiANCC2oXYrtBu1w6QAWozcaWqZKdjJw"
              ],
              "serviceEndpoint":{
                 "uri":"https://alice.aries.example.com:8081"
              }
           }
        ],
        "accept":[
           "didcomm/v2",
           "didcomm/aip2;env=rfc19",
           "didcomm/aip2;env=rfc587"
        ],
        "handshake_protocols":[
           "https://didcomm.org/didexchange/1.0"
        ]
     }
  }
```

Bob peut ensuite accepter l'invitation :

```
valonrexhepi@WINDOWSDESKTOP:~$ curl -k -X POST
"https://localhost:9082/outofband/accept-invitation" -H  "accept:
application/json" -H  "Content-Type: application/json" \
     -d '{
     "invitation":{
         "@id":"46226bf4-5ed3-49d5-a56b-868baafe26b9",
         "@type":"https://didcomm.org/out-of-band/1.0/invitation",
         "label":"Alice",
         "services":[
             {
                 "id":"26fb640e-749f-4277-9ad4-fe1f90686439",
                 "type":"DIDCommMessaging",
                 "recipientKeys":[
                     "did:key:z6LSr3Ti3jghy29UDiANCC2oXYrtBu1w6QAWozcaWqZKdjJw"
                 ],
                 "serviceEndpoint":{
                     "uri":"https://alice.aries.example.com:8081"
                 }
             }
         ],
         "accept":[
             "didcomm/v2",
             "didcomm/aip2;env=rfc19",
             "didcomm/aip2;env=rfc587"
         ],
         "handshake_protocols":[
```

```
            "https://didcomm.org/didexchange/1.0"
         ]
      },
      "my_label":"Bob"
 }'
{
    "connection_id":"872ef94f-8f60-43f5-85e4-b501360a20b3"
}
```

Si nous récupérons les connexions dont l'état est en "requête" chez Alice, nous obtenons ceci :

```
valonrexhepi@WINDOWSDESKTOP:~$ curl -k -X GET "https://localhost:8082/connections?
state=requested" -H  "accept: application/json" | python3 -mjson.tool
{
    "results": [
        {
            "ConnectionID": "06f3391a-13cd-4749-87e7-b26c3525445c",
            "State": "requested",
            "ThreadID": "188eae9c-05ce-4e27-b841-0e98c0a72b21",
            "ParentThreadID": "",
            "TheirLabel": "Bob",
            "TheirDID":
"did:peer:1zQmf4qtwEptTVvgCLEXSBhwYcjESjqxoTJxZsFV5aaF1iVC",
            "MyDID": "",
            "ServiceEndPoint": {
                "uri": ""
            },
            "RecipientKeys": null,
            "InvitationID": "46226bf4-5ed3-49d5-a56b-868baafe26b9",
            "InvitationDID": "",
            "Implicit": false,
            "Namespace": "their",
            "DIDCommVersion": "v1",
            "PeerDIDInitialState": ""
        }
    ]
}
```

Grâce au connectionID ci-dessus, nous pouvons accepter la requête depuis Alice :

```
valonrexhepi@WINDOWSDESKTOP:~$ curl -k -X POST
"https://localhost:8082/connections/06f3391a-13cd-4749-87e7-b26c3525445c/accept-
request" -H  "accept: application/json" | python3 -mjson.tool
{
    "their_did": "",
    "request_id": "",
    "connection_id": "06f3391a-13cd-4749-87e7-b26c3525445c",
    "updated_at": "0001-01-01T00:00:00Z",
    "created_at": "0001-01-01T00:00:00Z",
```

```
    "state": ""
}
```

Si depuis Alice et Bob nous récupérons encore une fois la liste des connexions, cette fois nous la voyons en "completed" :

```
valonrexhepi@WINDOWSDESKTOP:~$ curl -k -X GET "https://localhost:8082/connections"
-H  "accept: application/json" | python3 -mjson.tool
{
    "results": [
        {
            "ConnectionID": "06f3391a-13cd-4749-87e7-b26c3525445c",
            "State": "completed",
            "ThreadID": "188eae9c-05ce-4e27-b841-0e98c0a72b21",
            "ParentThreadID": "",
            "TheirLabel": "Bob",
            "TheirDID":
"did:peer:1zQmf4qtwEptTVvgCLEXSBhwYcjESjqxoTJxZsFV5aaF1iVC",
            "MyDID": "did:peer:1zQmdcZ7pBq7r9EQxACXvSb34WJ4ZDNRe2jhg3fKSuXkqxiH",
            "ServiceEndPoint": {
                "uri": "",
                "accept": [
                    "didcomm/v2"
                ]
            },
            "RecipientKeys": null,
            "InvitationID": "46226bf4-5ed3-49d5-a56b-868baafe26b9",
            "InvitationDID": "",
            "Implicit": false,
            "Namespace": "their",
            "DIDCommVersion": "v1",
            "PeerDIDInitialState": ""
        }
    ]
}
valonrexhepi@WINDOWSDESKTOP:~$ curl -k -X GET "https://localhost:9082/connections"
-H  "accept: application/json" | python3 -mjson.tool
{
    "results": [
        {
            "ConnectionID": "872ef94f-8f60-43f5-85e4-b501360a20b3",
            "State": "completed",
            "ThreadID": "188eae9c-05ce-4e27-b841-0e98c0a72b21",
            "ParentThreadID": "46226bf4-5ed3-49d5-a56b-868baafe26b9",
            "TheirLabel": "Alice",
            "TheirDID":
"did:peer:1zQmdcZ7pBq7r9EQxACXvSb34WJ4ZDNRe2jhg3fKSuXkqxiH",
            "MyDID": "did:peer:1zQmf4qtwEptTVvgCLEXSBhwYcjESjqxoTJxZsFV5aaF1iVC",
            "ServiceEndPoint": {
                "uri": "https://alice.aries.example.com:8081",
                "accept": [
```

```
                    "didcomm/v2",
                    "didcomm/aip2;env=rfc19",
                    "didcomm/aip2;env=rfc587"
                ]
            },
            "RecipientKeys": [
                "did:key:z6LSr3Ti3jghy29UDiANCC2oXYrtBu1w6QAWozcaWqZKdjJw"
            ],
            "InvitationID": "188eae9c-05ce-4e27-b841-0e98c0a72b21",
            "InvitationDID": "",
            "Implicit": false,
            "Namespace": "my",
            "DIDCommVersion": "v1",
            "PeerDIDInitialState": ""
        }
    ]
}
valonrexhepi@WINDOWSDESKTOP:~$
```

Faire un échange de présentation à travers le protocole de Present Proof

Pour faire un échange, nous devons obligatoirement avoir une connexion entre Alice et Bob, si ce n'est pas le cas il faut refaire le dernier exemple ci-dessus. Une fois que la connexion est faite, il faut noter les informations suivantes :

```
Alice DID : "did:peer:1zQmdcZ7pBq7r9EQxACXvSb34WJ4ZDNRe2jhg3fKSuXkqxiH"
Bob DID : "did:peer:1zQmf4qtwEptTVvgCLEXSBhwYcjESjqxoTJxZsFV5aaF1iVC"
```

Nous allons envoyer une requête de présentation depuis Alice en utilisant la méthode "/presentproof/send-request-presentation" :

```
valonrexhepi@WINDOWSDESKTOP:~$ curl -k -X POST
"https://localhost:8082/presentproof/send-request-presentation" \
 -H  "accept: application/json" \
 -H  "Content-Type: application/json" \
 -d '{
     "my_did":"did:peer:1zQmdcZ7pBq7r9EQxACXvSb34WJ4ZDNRe2jhg3fKSuXkqxiH",
     "their_did":"did:peer:1zQmf4qtwEptTVvgCLEXSBhwYcjESjqxoTJxZsFV5aaF1iVC",
     "request_presentation":{}
 }' | python3 -mjson.tool
{
    "piid": "67241eb8-bd84-4a36-8b41-3155100e1777"
}
```

Ici my_did représente le DID d'Alice et their_did celui de Bob. Nous pouvons maintenant accepter la requête de présentation depuis Bob avec la méthode "/presentproof/actions" :

```
valonrexhepi@WINDOWSDESKTOP:~$ curl -k -X GET
"https://localhost:9082/presentproof/actions" -H  "accept: application/json" |
python3 -mjson.tool
{
    "actions": [
        {
            "PIID": "67241eb8-bd84-4a36-8b41-3155100e1777",
            "Msg": {
                "@id": "67241eb8-bd84-4a36-8b41-3155100e1777",
                "@type": "https://didcomm.org/present-proof/2.0/request-
presentation",
                "~thread": {
                    "thid": "67241eb8-bd84-4a36-8b41-3155100e1777"
                }
            },
            "MyDID": "did:peer:1zQmf4qtwEptTVvgCLEXSBhwYcjESjqxoTJxZsFV5aaF1iVC",
            "TheirDID":
"did:peer:1zQmdcZ7pBq7r9EQxACXvSb34WJ4ZDNRe2jhg3fKSuXkqxiH"
        }
    ]
}
```

Pour accepter la requête depuis Bob, nous devons utiliser la méthode "/presentproof/{piid}/accept-request-presentation". Le payload encodé de présentation est :

```
{
    "@context":[
        "https://www.w3.org/2018/credentials/v1",
        "https://www.w3.org/2018/credentials/examples/v1"
    ],
    "holder":"did:example:ebfeb1f712ebc6f1c276e12ec21",
    "id":"urn:uuid:3978344f-8596-4c3a-a978-8fcaba3903c5",
    "type":[
        "VerifiablePresentation",
        "CredentialManagerPresentation"
    ],
    "verifiableCredential":null
}
```

Les informations de JWT sont :

```
{
    "alg":"none",
    "typ":"JWT"
}
```

Ainsi l'appel complet est :

```
valonrexhepi@WINDOWSDESKTOP:~$ curl -k -X POST
"https://localhost:9082/presentproof/67241eb8-bd84-4a36-8b41-3155100e1777/accept-
request-presentation" \
   -H  "accept: application/json" \
   -H  "Content-Type: application/json" \
   -d '{
      "presentation":{
         "presentations~attach":[
            {
               "lastmod_time":"0001-01-01T00:00:00Z",
               "data":{

"base64":"ZXlKaGJHY2lPaUp1YjI1bElpd2lkSGx3SWpvaVNsZFVJbjAuZXlKcGMzTWlPaUprYVdRNlpY
aGhiWEJzWlRwbFltWmxZakZtTnpFeVpXSmpObVl4WXpJM05tVXhNbVZqTWpFaUxDSnFkR2tpT2lKMWNtND
ZkWFZwWkRvek9UYzRNelEwWmkwNE5UazJMVFJqTTJFdFlUazNPQzA0Wm1OaFltRXpPVEF6WXpVaUxDSjJj
Q0k2ZXlKQVkyOXVkR1Y0ZENJNld5Sm9kSFJ3Y3pvdkwzZDNkeTUzTXk1dmNtY3ZNakF4T0M5amNtVmtaVz
UwYVdGc2N5OTJNU0lzSW1oMGRIQnpPaTh2ZDNkM0xuY3pMbTl5Wnk4eU1ERTRMMk55WldSbGJuUnBZV3h6
TDJWNFlXMXdiR1Z6TDNZeElsMHNJbWh2YkdSbGNpSTZJbVJwWkRwbGVHRnRjR3hsT21WaVptVmlNV1kzTV
RKbFltTTJaakZqTWpjMlpURXlaV015TVNJc0ltbGtJam9pZFhKdU9uVjFhV1E2TXprM09ETTBOR1l0T0RV
NU5pMDBZek5oTFdFNU56Z3RPR1pqWVdKaE16a3dNMk0xSWl3aWRIbHdaU0k2V3lKV1pYSnBabWxoWW14bF
VISmxjMlZ1ZEdGMGFXOXVJaXdpUTNKbFpHVnVkR2xoYkUxaGJtRm5aWEpRY21WelpXNTBZWFJwYjI0aVhT
d2lkbVZ5YVdacFlXSnNaVU55WldSbGJuUnBZV3dpT201MWJHeDlmUS4="
               }
            }
         ]
      }
   }' | python3 -mjson.tool
{}
```

Depuis Alice, nous allons accepter la présentation. La première étape est de connaître le PIID, nous faisons alors appel à la méthode "/presentproof/actions" :

```
curl -k -X GET "https://localhost:9082/presentproof/actions" -H  "accept:
application/json"
CELA NE FONCTIONNE PAS
```

Ensuite, nous appelons la méthode "/presentproof/{piid}/accept-presentation" depuis Alice en donnant un nom à la présentation :

```
valonrexhepi@WINDOWSDESKTOP:~$ curl -k -X POST
"https://localhost:8082/presentproof/67241eb8-bd84-4a36-8b41-3155100e1777/accept-
presentation" \
 -H  "accept: application/json" \
 -H  "Content-Type: application/json" \
 -d '{
    "names":[
       "demo-presentation"
    ]
```

```
 }' | python3 -mjson.tool
{}
```

Finalement, nous pouvons vérifier que la présentation a été sauvegardée avec la méthode "/verifiable/presentations" :

```
valonrexhepi@WINDOWSDESKTOP:~$ curl -k -X GET
"https://localhost:8082/verifiable/presentations" -H  "accept: application/json" |
python3 -mjson.tool
{
    "result": [
        {
            "name": "demo-presentation",
            "id": "urn:uuid:3978344f-8596-4c3a-a978-8fcaba3903c5",
            "context": [
                "https://www.w3.org/2018/credentials/v1",
                "https://www.w3.org/2018/credentials/examples/v1"
            ],
            "type": [
                "VerifiablePresentation",
                "CredentialManagerPresentation"
            ],
            "subjectId": "did:example:ebfeb1f712ebc6f1c276e12ec21",
            "my_did": "did:peer:1zQmdcZ7pBq7r9EQxACXvSb34WJ4ZDNRe2jhg3fKSuXkqxiH",
            "their_did":
"did:peer:1zQmf4qtwEptTVvgCLEXSBhwYcjESjqxoTJxZsFV5aaF1iVC"
        }
    ]
}
```

Émettre des crédentiels à travers le protocole Issue Credential

Comme pour l'exercice d'avant, il est nécessaire d'avoir une connexion entre Alice et Bob pour continuer. Notons que dans ce lab nous utilisons parfois l'argument "-k", celui-ci permet d'éviter la vérification sécurisée de la requête car durant la réalisation de ce lab, nous avons rencontré un problème avec les certificats. Si la connexion n'existe pas, il faut la créer via le "Créer une connexion DID à travers le protocole out-of-band".

Nous disposons des informations suivantes :

```
Alice DID : "did:peer:1zQmdcZ7pBq7r9EQxACXvSb34WJ4ZDNRe2jhg3fKSuXkqxiH"
Bob DID : "did:peer:1zQmf4qtwEptTVvgCLEXSBhwYcjESjqxoTJxZsFV5aaF1iVC"
```

Depuis Alice, nous appelons la méthode "/issuecredential/send-offer" :

```
valonrexhepi@WINDOWSDESKTOP:~$ curl -k -X POST
"https://localhost:8082/issuecredential/send-offer" \
   -H  "accept: application/json" \
```

```
    -H  "Content-Type: application/json" \
    -d '{
        "my_did": "did:peer:1zQmdcZ7pBq7r9EQxACXvSb34WJ4ZDNRe2jhg3fKSuXkqxiH",
        "their_did": "did:peer:1zQmf4qtwEptTVvgCLEXSBhwYcjESjqxoTJxZsFV5aaF1iVC",
        "offer_credential": {}
    }' | python3 -mjson.tool
{
    "piid": "36542a8b-f379-4a84-a876-c0683e9c7be1"
}
```

Ici my_did est le DID d'Alice et their_did est celui de Bob. Ensuite depuis Bob, nous pouvons accepter l'offre en effectuant d'abord la méthode "/issuecredential/actions" pour connaître le PIID :

```
valonrexhepi@WINDOWSDESKTOP:~$ curl -k -X GET
"https://localhost:9082/issuecredential/actions" -H  "accept: application/json" |
python3 -mjson.tool | python3 -mjson.tool
{
    "actions": [
        {
            "PIID": "36542a8b-f379-4a84-a876-c0683e9c7be1",
            "Msg": {
                "@id": "36542a8b-f379-4a84-a876-c0683e9c7be1",
                "@type": "https://didcomm.org/issue-credential/2.0/offer-
credential",
                "~thread": {
                    "thid": "36542a8b-f379-4a84-a876-c0683e9c7be1"
                }
            },
            "MyDID": "did:peer:1zQmf4qtwEptTVvgCLEXSBhwYcjESjqxoTJxZsFV5aaF1iVC",
            "TheirDID":
"did:peer:1zQmdcZ7pBq7r9EQxACXvSb34WJ4ZDNRe2jhg3fKSuXkqxiH"
        }
    ]
}
```

Et lorsque nous avons le PIID, nous utilisons la méthode "/issuecredential/{piid}/accept-offer" pour accepter l'offre :

```
valonrexhepi@WINDOWSDESKTOP:~$ curl -k -X POST
"https://localhost:9082/issuecredential/36542a8b-f379-4a84-a876-
c0683e9c7be1/accept-offer" -H  "accept: application/json" | python3 -mjson.tool
{}
```

Depuis Alice, nous acceptons la requête en appelant d'abord la méthode "/issuecredential/actions" pour connaître le PIID :

```
valonrexhepi@WINDOWSDESKTOP:~$ curl -k -X GET
"https://localhost:8082/issuecredential/actions" -H  "accept: application/json" |
python3 -mjson.tool
{
    "actions": [
        {
            "PIID": "36542a8b-f379-4a84-a876-c0683e9c7be1",
            "Msg": {
                "@id": "d8351891-45fc-4d8f-9ea7-adb4e6f82e56",
                "@type": "https://didcomm.org/issue-credential/2.0/request-
credential",
                "~thread": {
                    "thid": "36542a8b-f379-4a84-a876-c0683e9c7be1"
                }
            },
            "MyDID": "did:peer:1zQmdcZ7pBq7r9EQxACXvSb34WJ4ZDNRe2jhg3fKSuXkqxiH",
            "TheirDID":
"did:peer:1zQmf4qtwEptTVvgCLEXSBhwYcjESjqxoTJxZsFV5aaF1iVC"
        }
    ]
}
```

Nous utilisons ensuite ce PIID pour appeler la méthode "/issuecredential/{piid}/accept-request" pour accepter la requête et envoyer les crédentiels :

```
valonrexhepi@WINDOWSDESKTOP:~$ curl -k -X POST
"https://localhost:8082/issuecredential/36542a8b-f379-4a84-a876-
c0683e9c7be1/accept-request" \
   -H  "accept: application/json" \
   -H  "Content-Type: application/json" \
   -d '{
      "issue_credential":{
         "credentials~attach":[
            {
               "lastmod_time":"0001-01-01T00:00:00Z",
               "data":{
                  "json":{
                     "@context":[
                        "https://www.w3.org/2018/credentials/v1",
                        "https://www.w3.org/2018/credentials/examples/v1"
                     ],
                     "credentialSubject":{
                        "id":"sample-credential-subject-id"
                     },
                     "id":"http://example.edu/credentials/1872",
                     "issuanceDate":"2010-01-01T19:23:24Z",
                     "issuer":{
                        "id":"did:example:76e12ec712ebc6f1c221ebfeb1f",
                        "name":"Example University"
                     },
                     "referenceNumber":83294847,
```

```
                    "type":[
                       "VerifiableCredential",
                       "UniversityDegreeCredential"
                    ]
                 }
              }
           }
        ]
     }
  }' | python3 -mjson.tool
{}
```

Bob peut maintenant accepter le crédentiel, pour ce faire il doit appeler la méthode "/issuecredential/actions" pour connaître le PIID :

```
valonrexhepi@WINDOWSDESKTOP:~$ curl -k -X GET
"https://localhost:9082/issuecredential/actions" -H  "accept: application/json" |
python3 -mjson.tool
{
    "actions": []
}
CELA NE FONCTIONNE PAS
```

Avec le PIID, la méthode "/issuecredential/{piid}/accept-credential" est ensuite appelée pour accepter les crédentiels :

```
valonrexhepi@WINDOWSDESKTOP:~$ curl -k -X POST
"https://localhost:9082/issuecredential/36542a8b-f379-4a84-a876-
c0683e9c7be1/accept-credential" \
   -H  "accept: application/json" \
   -H  "Content-Type: application/json" \
   -d '{
      "names":[
         "demo-credential"
      ]
   }' | python3 -mjson.tool
{
    "code": 8004,
    "message": "get transitional payload: store get: failed to get DB entry: data
not found"
}
CELA NE FONCTIONNE PAS
```

Finalement, Bob peut vérifier que le crédentiel a bien été sauvegardé :

```
valonrexhepi@WINDOWSDESKTOP:~$ curl -k -X GET
"https://localhost:9082/verifiable/credential/name/demo-credential" -H  "accept:
```

```
application/json" | python3 -mjson.tool
{
    "code": 6004,
    "message": "get vc by name : fetch credential id based on name : failed to get
DB entry: data not found"
}
CELA NE FONCTIONNE PAS
```

Pour aller plus loin

Le lab nous indique également que nous pouvons passer du temps à faire plus d'exploration en se tournant vers la méthode did:orb ainsi que pour l'utilisation du Wallet Opensource TrustBloc qui ensemble forme un écosystème d'utilisation pour Aries Go.

HyperLedger Aries - Analyse de l'architecture

Ce document synthétise des concepts clés de l'architecture d'HyperLedger Aries.

Synthèse

L'écosystème HyperLedger SSI

Pour comprendre les différentes variantes d'Aries, il faut comprendre l'écosystème SSI d'HyperLedger. Il est composé de HyperLedger Indy, HyperLedger Ursa et HyperLedger Aries.

HyperLedger Indy est une framework blockchain pour l'identité. La fondation Sovrin est l'un des contributeurs au code. C'est un Ledger Distribué pour l'identité décentralisée. Indy inclut également des concepts comme le ZKP, des identifiers décentralisés, un SDK et une implémentation d'un ledger distribué avec permission publique.

HyperLedger Ursa, c'est simplement une framework dédiée à la cryptographie. À la base, les parties cryptographiques faisaient partie d'Indy mais la communauté s'est rendu compte que celles-ci pouvaient servir à d'autres projets, ils ont donc décidé de sortir le package et de l'implémenter à part, dans HyperLedger Ursa.

HyperLedger Aries part de l'idée d'une framework SSI dont les agents peuvent utiliser des DID et Verifiable Credentials d'écosystèmes différents et multiples. Aries est un toolkit, il utilise l'interaction en peer-to-peer entre les agents. En utilisant le channel standardisé, les credentials vérifiables peuvent être échangés en se basant sur des DID qui se trouvent sur différents ledgers (basés sur Indy ou non).

Les variantes d'Aries

Il y a plusieurs variantes de la framework Aries :

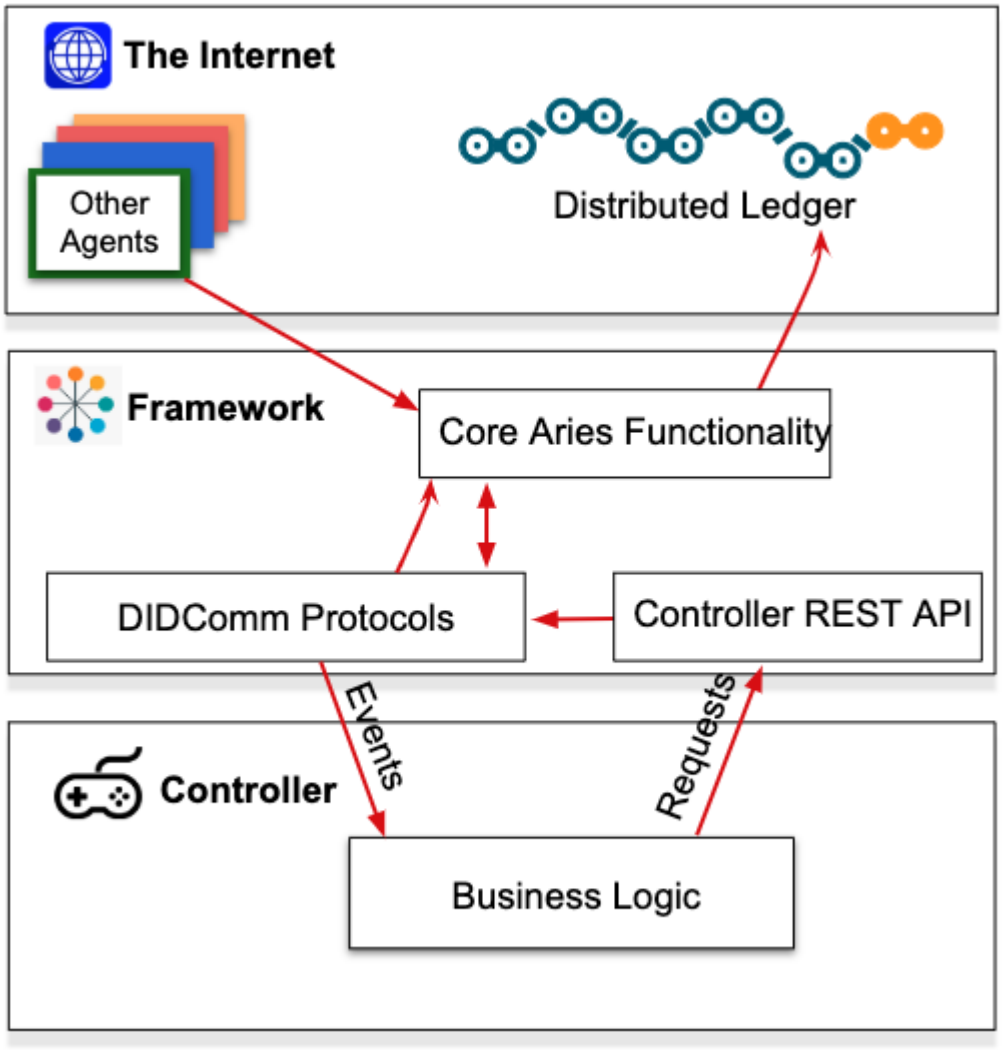
- Aries Cloud Agent Python est pour tous les agents non mobiles. ACA-Py ainsi qu'un contrôleur s'exécute ensemble et communique à travers une interface HTTP. Le contrôleur peut être écrit dans n'importe quel langage, ACA-Py intègre le SDK Indy.
- Aries Framework .NET est utilisé pour le développement mobile et côté-serveur. Le contrôleur peut également être écrit dans n'importe quel langage qui permet d'intégrer la framework en tant que librairie dans le contrôleur. Il intègre également le SDK Indy.
- Aries Framework Go est une framework pure en Go. Elle utilise une architecture similaire à ACA-Py. Il y a une interface HTTP qui communique avec un contrôleur. La framework n'intègre pas pour l'instant l'Indy SDK. La mise en oeuvre de credentials vérifiables basés sur le golang est en développement.
- Aries Static Agent Python est un agent configurable qui n'utilise pas de stockage persistent.
- Il existe également des versions Ruby et Javascript

Il existe également des outils de développement à disposition :

- Aries-toolbox qui simplifie la visualisation de l'interaction entre agents
- Aries-protocol-test-suite permet de tester si notre agent est compatible Aries

L'architecture d'Aries

Les deux versions les plus intéressantes pour notre projet sont la version ACA-Py ainsi que la version golang AF-Go. Comme nous l'avons dit précédemment, l'architecture utilisée pour ces deux versions est la même. Voilà un schéma représentatif pour l'architecture d'ACA-Py.



Cependant, la version AF-Go est plus flexible et indépendante que celle en Python. Le projet suivant qui est le respository officiel de la framework en Go, peut être utilisé de plusieurs façons différentes :

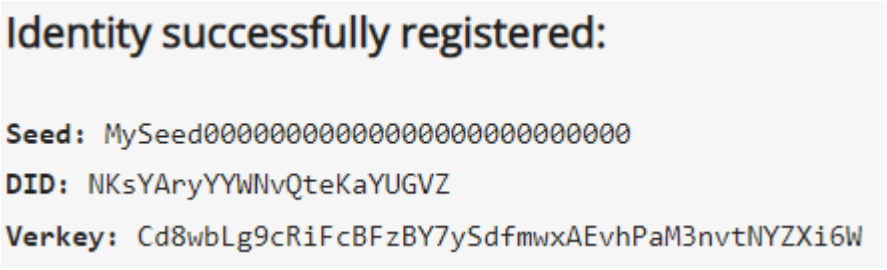
- en tant que framework pour construire des agents Aries en Go
- en tant qu'agent serveur et ensuite même architecture et utilisation qu'ACA-Py
- en tant que framework Go pour le développement Javascript
- en tant que framework Go pour le développement Mobile

Comment fonctionne les DID sur Aries Go ?

Les différents standard, protocoles et spécifications utilisés par l'Aries Framework Go peuvent être trouvés ici. Ainsi, nous allons dans cette section parler des DID.

Aries utilise les spécifications du W3C pour résoudre et déconstruire les DID. Si la résolution pour un DID n'est pas implémentée dans la Framework, un universal resolver est utilisé pour résoudre les DID. Il est également important de mentionner que les travaux de la Decentralized Identity Foundation sont globalement utilisés dans le traitement de DID par la communauté SSI.

Dans un lab précédent, nous avons vu que lorsque nous utilisons la blockchain Sovrin le DID est écrit et récupéré depuis le ledger.



La Framework inclut également le protocole DIDComm Messaging qui permet une communication sécurisée à travers une couche de protocole construite en dessus des DID.

Comment fonctionne les credentials sur Aries Go ?

Le traitement des credentials se fait également à travers les standards du W3C. Pour les opérations du modèle de données des credentials vérifiables, c'est le standard suivant qui est utilisé. Pour la vérification des credentials et de la présentation, c'est le standard suivant.

De la même façon, les différents credentials, proofs et présentations sont écrits et récupérés depuis le ledger. Voilà un exemple d'initialisation d'un schéma qui est ensuite utilisé comme credential.

```
await faber_container.initialize(
            schema_name="degree schema",
            schema_attrs=[
                "name",
                "date",
                "degree",
                "grade",
            ],
        )
```

Ici est exemple de son utilisation.

```
#28.1 Received proof of education, check claims
Acme       | name: Alice Smith
Acme       | date: 2018-05-28
Acme       | degree: Maths
Acme       | schema_id: KcV47U9edQi2NbjkJuJzLF:2:degree schema:32.95.31
Acme       | cred_def_id
KcV47U9edQi2NbjkJuJzLF:3:CL:131995:faber.agent.degree_schema
    (1) Issue Credential
```

L'exemple complet de l'émission d'un credential peut être trouvé dans le lab 6.

Références

3BLUE1BROWN. (2017, juillet 7). But how does bitcoin actually work ? Récupérée le 28 juin 2022, à partir de https://www.youtube.com/watch?v=bBC-nXj3Ng4

AGARWAL, A. (2022, février 11). Which Is The Most Decentralized Cryptocurrency ? Récupérée le 11 juin 2022, à partir de https://medium.com/coinmonks/which-is-the-most-decentralized-cryptocurrency-c446d6cedc9e

AIR FRANCE. (s. d.). ENREGISTREMENT. Récupérée le 11 juillet 2022, à partir de https : //wwws.airfrance.fr/information/aeroport/enregistrement

ALLEN, C. (2016, avril 25). The Path to Self-Sovereign Identity. Récupérée le 31 mai 2022, à partir de http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html

ANTONOPOULOS, A. (2014). *Mastering Bitcoin*. O'Reilly Media, Inc.

APPLE. (2022a, mai 5). Apple, Google et Microsoft s'engagent à étendre l'intégration de la norme FIDO pour accélérer la mise à disposition des connexions sans mot de passe. Récupérée le 30 juin 2022, à partir de https://www.apple.com/chfr/newsroom/2022/05/apple-google-and-microsoft-commit-to-expanded-support-for-fido-standard/

APPLE. (2022b, juin 6). Wallet. Récupérée le 7 juin 2022, à partir de https://www.apple.com/wallet/

APPLE. (2022c, juin 6). WWDC 2022 - June 6 | Apple. Récupérée le 30 juin 2022, à partir de https://www.youtube.com/watch?v=q5D55G7Ejs8

BARBEY, G. (2022, juin 30). e-ID : tout ce qu'il faut savoir sur le nouveau projet des autorités. Récupérée le 4 juillet 2022, à partir de https://www.heidi.news/innovation-solutions/e-id-tout-ce-qu-il-faut-savoir-sur-le-nouveau-projet-des-autorites

BEDOYA, J. P. (2022, mai 27). Who Are We in the Metaverse, and How Do We Prove It ? Récupérée le 31 mai 2022, à partir de https://www.coindesk.com/layer2/metaverseweek/2022/05/27/who-are-we-in-the-metaverse-and-how-do-we-prove-it/

BOLTE, P. (2021). *Self-sovereign Identity : Development of an Implementation-basedEvaluation Framework for Verifiable Credential SDKs* (mém. de mast., Technische Hochschule Brandenburg).

BUTERIN, V. (2022, janvier 26). Soulbound. Récupérée le 1 juillet 2022, à partir de https : //vitalik.ca/general/2022/01/26/soulbound.html

CENTRE NATIONAL POUR LA CYBERSÉCURITÉ. (2020, décembre 8). Prudence : un nombre croissant de PME victimes de rançongiciels. Récupérée le 27 mai 2022, à partir de https://www.ncsc.admin.ch/ncsc/fr/home/aktuell/news/news-archiv/sicherheitsrisiko-durch-ransomware.html

CHOHAN, U. W. (2021). The Double Spending Problem and Cryptocurrencies. doi :10.2139/ssrn.3090174

DÉLÈZE IVAN, A. (2020). *Hacking Sovereign Identity* (mém. de mast., École Polytechnique Fédérale de Lausanne).

DÉPARTEMENT FÉDÉRAL DE JUSTICE ET POLICE. (2021, mars 7). Identité électronique : la loi sur l'e-ID. Récupérée le 24 mai 2022, à partir de https://www.ejpd.admin.ch/ejpd/fr/home/themes/abstimmungen/bgeid.html#:~:text=La%5C%20nouvelle%5C%20loi%5C%20sur%5C%20l,une%5C%20e%5C%2DID%5C%20est%5C%20facultative.

DIGICONOMIST. (s. d.-a). Bitcoin Energy Consumption Index. Récupérée le 13 juin 2022, à partir de https://digiconomist.net/bitcoin-energy-consumption

DIGICONOMIST. (s. d.-b). Ethereum Energy Consumption Index. Récupérée le 11 juin 2022, à partir de https://digiconomist.net/ethereum-energy-consumption/

DU, L., HO, A. T. & CONG, R. (2020). Perceptual hashing for image authentication : A survey. *Signal Processing : Image Communication*, *81*, 115713. doi :https://doi.org/10.1016/j.image.2019.115713

EMSISOFT MALWARE LAB. (2019, décembre 12). The State of Ransomware in the US : Report and Statistics 2019. Récupérée le 27 mai 2022, à partir de https://blog.emsisoft.com/en/34822/the-state-of-ransomware-in-the-us-report-and-statistics-2019/

ETHEREUM. (s. d.-a). ethereum.org the portal into the world of Ethereum. Récupérée le 11 juin 2022, à partir de https://ethereum.org/

ETHEREUM. (s. d.-b). Non-fungible tokens (NFT). Récupérée le 1 juillet 2022, à partir de https://ethereum.org/en/nft/

ETHUB. (s. d.). Welcome to EthHub. Récupérée le 11 juin 2022, à partir de https://docs.ethhub.io/

FIDO ALLIANCE. (s. d.). FIDO Alliance website. Récupérée le 30 juin 2022, à partir de https://fidoalliance.org/

GABRIELLA, L., TAIJA, K. & PEKKA, A. (2021). Self-Sovereign Identity Ecosystems : Benefits and Challenges. *12th Scandinavian Conference on Information Systems. 10.* Récupérée le 7 juin 2022, à partir de https://aisel.aisnet.org/scis2021/10

GISOLFI, D. (2018, juin 13). Self-sovereign identity : Why blockchain ? Récupérée le 8 juin 2022, à partir de https://www.ibm.com/blogs/blockchain/2018/06/self-sovereign-identity-why-blockchain/

GOLDREICH, O. (2003, novembre 1). Showing without Telling. Récupérée le 6 juin 2022, à partir de https://wis-wander.weizmann.ac.il/math-computer-science/showing-without-telling

GRIMES, R. (2021). *Hacking Multifactor Authentication*. Wiley.

HARDMAN, D. (2019, mars 1). What if I lose my phone ? Récupérée le 5 juillet 2022, à partir de https://sovrin.org/wp-content/uploads/2019/03/What-if-someone-steals-my-phone-110319.pdf

HOOD, D. (2022, avril 13). The Top 10 Most Expensive NFTs. Récupérée le 1 juillet 2022, à partir de https://www.business2community.com/nft/most-expensive-nft

HYPERLEDGER. (s. d.). Hyperledger Architecture, Volume 1. Récupérée le 14 juin 2022, à partir de https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf

HYPERLEDGER FOUNDATION. (s. d.-a). Aries. Récupérée le 29 juin 2022, à partir de https://github.com/hyperledger/aries

HYPERLEDGER FOUNDATION. (s. d.-b). Aries Framework for .NET. Récupérée le 29 juin 2022, à partir de https://github.com/hyperledger/aries-framework-dotnet

HYPERLEDGER FOUNDATION. (s. d.-c). Aries Framework Go. Récupérée le 29 juin 2022, à partir de https://github.com/hyperledger/aries-framework-go

HYPERLEDGER FOUNDATION. (s. d.-d). Hyperledger Aries. Récupérée le 29 juin 2022, à partir de https://www.hyperledger.org/use/ariesk

HYPERLEDGER FOUNDATION. (s. d.-e). Hyperledger Aries. Récupérée le 29 juin 2022, à partir de https://wiki.hyperledger.org/display/aries

HYPERLEDGER FOUNDATION. (s. d.-f). Hyperledger Aries Cloud Agent - Python. Récupérée le 29 juin 2022, à partir de https://github.com/hyperledger/aries-cloudagent-python

HYPERLEDGER INDY. (s. d.). Hyperledger Indy Documentation. Récupérée le 14 juin 2022, à partir de https://indy.readthedocs.io/en/latest/

IBM. (2022, avril 19). TCP/IP protocols. Récupérée le 27 mai 2022, à partir de https://www.ibm.com/docs/en/aix/7.2?topic=protocol-tcpip-protocols

IDENTITY FOUNDATION. (s. d.-a). Identity Foundation Website. Récupérée le 13 juin 2022, à partir de https://identity.foundation/

IDENTITY FOUNDATION. (s. d.-b). ION Repository. Récupérée le 13 juin 2022, à partir de https://github.com/decentralized-identity/ion

IOTA. (s. d.-a). IOTA Foundation Blog. Récupérée le 27 juin 2022, à partir de https://blog.iota.org/

IOTA. (s. d.-b). IOTA Foundation Website. Récupérée le 13 juin 2022, à partir de https://www.iota.org/

IOTA SERVICES. (s. d.). What is IOTA ? Récupérée le 13 juin 2022, à partir de https://www.iota-services.com/what-is-iota

IOTA WIKI. (s. d.). IOTA Foundation Wikipedia. Récupérée le 13 juin 2022, à partir de https://wiki.iota.org/

LINUX FOUNDATION. (s. d.). Becoming a Hyperledger Aries Developer. Récupérée le 29 juin 2022, à partir de https://learnthings.online/course/2020/03/06/becoming-a-hyperledger-aries-developer

MARFICE, C. (2020, septembre 1). The Evolution of Ecommerce [10-50 Years] [Timeline]. Récupérée le 31 mai 2022, à partir de https://www.plytix.com/blog/evolution-of-ecommerce-timeline#:~:text=In%5C%202020%5C%2C%5C%20global%5C%20ecommerce%5C%20sales,on%5C%20mobile%5C%20devices%5C%20in%5C%202021.

MATTR. (s. d.). MATTR website. Récupérée le 29 juin 2022, à partir de https://mattr.global/

MATTR LEARN. (s. d.-a). Create an ION DID. Récupérée le 27 juin 2022, à partir de https://learn.mattr.global/tutorials/dids/did-ion

MATTR LEARN. (s. d.-b). MATTR Learning Platform. Récupérée le 29 juin 2022, à partir de https://learn.mattr.global/

MORGAN, S. (2015, novembre 24). IBM's CEO On Hackers : 'Cyber Crime Is The Greatest Threat To Every Company In The World'. Récupérée le 27 mai 2022, à partir de https://www.forbes.com/sites/stevemorgan/2015/11/24/ibms-ceo-on-hackers-cyber-crime-is-the-greatest-threat-to-every-company-in-the-world/?sh=112bb18973f0

PARTISIA BLOCKCHAIN. (s. d.-a). Partisia Blockchain Documentation. Récupérée le 14 juin 2022, à partir de https://partisiablockchain.gitlab.io/documentation/index.html

PARTISIA BLOCKCHAIN. (s. d.-b). Partisia Blockchain Website. Récupérée le 14 juin 2022, à partir de https://partisiablockchain.com/

PARTISIA BLOCKCHAIN. (2021, mars 18). Partisia Blockchain. Récupérée le 14 juin 2022, à partir de https://partisia.com/wp-content/uploads/2021/03/Partisia_WhitePaper_1_06.pdf

PRAKASH, P., DING, J., LI, H., ERRAPOTU, S. M., PEI, Q. & PAN, M. (2020). Privacy Preserving Facial Recognition Against Model Inversion Attacks, 1-6. doi :10.1109/GLOBECOM42002.2020.9322508

PREUKSCHAT, A. & REED, D. [D.]. (2021). *Self-Sovereign Identity*. Manning.

RTS. (2022, juin 29). La population sera plus impliquée dans la future identité numérique. Récupérée le 29 juin 2022, à partir de https://www.rts.ch/info/suisse/13209089-la-population-sera-plus-impliquee-dans-la-future-identite-numerique.html

RUFF, T. (2018, avril 24). The Three Models of Digital Identity Relationships. Récupérée le 30 mai 2022, à partir de https://medium.com/evernym/the-three-models-of-digital-identity-relationships-ca0727cb5186

SCION. (s. d.). SCION's Website. Récupérée le 30 mai 2022, à partir de https://scion-architecture.net/

SHUAIB, M., ALAM, S., SHABBIR ALAM, M. & SHAHNAWAZ NASIR, M. (2021). Self-sovereign identity for healthcare using blockchain. *Materials Today : Proceedings*. doi :https://doi.org/10.1016/j.matpr.2021.03.083

SIMONS, A. (2019, mai 13). Toward scalable decentralized identifier systems. Récupérée le 13 juin 2022, à partir de https://techcommunity.microsoft.com/t5/azure-active-directory-identity/toward-scalable-decentralized-identifier-systems/ba-p/560168#

SORI ABBASZADEH, A. (2019, août 16). Green IOTA, Measuring IOTA PoW 's Energy Consumption and Comparing with other Payment Systems. Récupérée le 13 juin 2022, à partir de https://medium.com/@a.abbaszadeh.s/measuring-iota-pow-s-energy-consumption-and-comparing-with-other-payment-systems-413f4de50274

SOVRIN. (s. d.). Sovrin Website. Récupérée le 14 juin 2022, à partir de https://sovrin.org/

SOVRIN. (2019, décembre 4). Sovrin Glossary V3. Récupérée le 7 juin 2022, à partir de https://sovrin.org/wp-content/uploads/Sovrin-Glossary-V3.pdf

SWISSCOM. (s. d.). L'Internet ultra-performant et sécurisé sous votre contrôle. Récupérée le 30 mai 2022, à partir de https://www.swisscom.ch/fr/business/enterprise/offre/wireline/scion.html

THOMAS JEFFERSON UNIVERSITY. (2016, novembre 22). FROM ARPANET TO WORLD WIDE WEB : AN INTERNET HISTORY TIMELINE. Récupérée le 27 mai 2022, à partir de https://online.jefferson.edu/business/internet-history-timeline/

TOBIN, A. & REED, D. [Drummond.]. (2016, septembre 29). The Inevitable Rise of Self-Sovereign Identity. Récupérée le 31 mai 2022, à partir de https://sovrin.org/wp-content/uploads/2017/06/The-Inevitable-Rise-of-Self-Sovereign-Identity.pdf

TRUSTOVERIP. (2022, février 21). THE TOIP FOUNDATION RELEASES ITS FIRST OFFICIAL GOVERNANCE SPECIFICATIONS. Récupérée le 4 juillet 2022, à partir de https://trustoverip.org/news/2022/02/01/the-toip-foundation-releases-its-first-official-governance-specifications/

VARSHNEY, N. (2018, mai 24). Why Proof-of-work isn't suitable for small cryptocurrencies. Récupérée le 11 juillet 2022, à partir de https://thenextweb.com/news/proof-work-51-percent-attacks

W3C. (2021, août 3). Decentralized Identifiers (DIDs) v1.0. Récupérée le 7 juin 2022, à partir de https://w3c.github.io/did-core/

W3C. (2022a, mars 4). Universal Wallet 2020. Récupérée le 7 juin 2022, à partir de https://w3c-ccg.github.io/universal-wallet-interop-spec/

W3C. (2022b, mars 3). Verifiable Credentials Data Model v1.1. Récupérée le 6 juin 2022, à partir de https://www.w3.org/TR/vc-data-model/

WENG, L. & PRENEEL, B. (2007). Attacking Some Perceptual Image Hash Algorithms, 879-882. doi :10.1109/ICME.2007.4284791

WESTON, G. (2022, juin 29). SoulBound Tokens- The Non-Transferable NFTs. Récupérée le 1 juillet 2022, à partir de https://101blockchains.com/soulbound-tokens/

WILSER, J. (2020, octobre 1). Self-Sovereign Identity Explained. Récupérée le 31 mai 2022, à partir de https://www.coindesk.com/policy/2020/10/01/self-sovereign-identity-explained/

WORLD BANK GROUP. (2017). Distributed Ledger Technology and Blockchain. Récupérée le 8 juin 2022, à partir de https://documents1.worldbank.org/curated/en/177911513714062215/pdf/122140-WP-PUBLIC-Distributed-Ledger-Technology-and-Blockchain-Fintech-Notes.pdf

YCHARTS. (s. d.-a). Bitcoin Average Transaction Fee. Récupérée le 13 juin 2022, à partir de https://ycharts.com/indicators/bitcoin_average_transaction_fee

YCHARTS. (s. d.-b). Ethereum Average Transaction Fee. Récupérée le 11 juin 2022, à partir de https://ycharts.com/indicators/ethereum_average_transaction_fee

ZAHARIA, A. (2022, février 22). 300+ Terrifying Cybercrime and Cybersecurity Statistics (2022 EDITION). Récupérée le 27 mai 2022, à partir de https://www.comparitech.com/vpn/cybersecurity-cyber-crime-statistics-facts-trends/

Acronymes

ARPANET Advanced Research Projects Agency Network. 4, 5

CERN Conseil européen pour la recherche nucléaire. 5

DARPA Defense Advanced Research Projects Agency. 4

DFJP Département fédéral de justice et police. ii, 67, 71, 111

DID Decentralized Identifier. vi, 24, 33, 34, 47–49, 55–57, 59, 60, 74, 77, 79, 93, 99

DIF Decentralized Identity Foundation. 49, 53, 54

DL Distributed Ledger. 38, 39, 42, 50, 52, 56–58, 64, 66, 74, 76–78, 111

DLT Distributed Ledger Technology. viii, 4, 12, 24, 34, 38, 39, 47, 48, 50, 53, 54

DNS Domain Name System. 5

IOT Internet of Things. 12, 57, 66

IP Internet Protocol. 81

JSON-LD JSON for Linking Data. 26, 33

LPD Loi fédérale sur la protection des données. 36, 64

NFT Non-Fungible Token. 64–66

PoC Proof of Concept. 74, 79, 80, 87–92, 94, 96, 99, 103, 110, 111

RBFT Redundant Byzantine Fault Tolerance. 52

RGPD Règlement général sur la protection des données. 36, 50, 52, 64

SCION Scalability, Control, and Isolation On Next-Generation Networks. 6

SSI Self-Sovereign Identity. iii, vi, viii, 4, 6, 7, 11–17, 24, 27, 28, 32, 33, 35–38, 40, 47, 48, 55, 57–59, 65–72, 74, 111, 112

TCP/IP Transmission Control Protocol/Internet Protocol. 4, 5, 34

URL Uniform Resource Locator. 34, 76, 78

VC Verifiable Credentials. 24–28, 31, 37, 47, 55, 57, 59, 77, 88

ZKP Zero-Knowledge Proofs. 27, 48–53, 55, 65, 66

Glossaire

Agile Les méthodologies ou pratiques dites "agiles" vont mettre en avant les relations humaines entre les membres d'une équipe et leurs clients. Au centre de celles-ci, nous allons retrouver des valeurs comme la flexibilité, l'amélioration continue ou encore la communication. Elles sont le plus souvent en lien avec les domaines informatiques mais peuvent être utilisées dans n'importe quel autre domaine. 2

Anti-Money Laudering C'est un processus utilisé pour combattre le blanchiment d'argent. 15

Application Programming Interface Une interface de programmation d'application est un ensemble de composants qui permet à un logiciel d'offrir des services à d'autres logiciels. 57, 58, 60, 61, 81, 84, 86, 94, 95, 97, 99

Authentification multifacteur Un facteur d'authentification va permettre de confirmer son identité sur un service auquel nous souhaitons nous connecter. Une authentification multifacteur va combiner plusieurs types de facteurs différents, par exemple un mot de passe et un code reçu par SMS, pour renforcer la sécurité de cette connexion. 62, 73

Claim C'est une information sur un sujet. 26–28

Classe En programmation, une classe est un ensemble de code commun à un objet. 90

Cloud Ce terme désigne l'ensemble des solutions de stockage distant, comme des serveurs par exemple. 57

Cybercrime C'est une infraction pénale commise sur un système informatique connecté à un réseau. ii, 6, 36, 70

Digest C'est la représentation alpha-numérique, de taille fixe, du contenu d'un message. Elle est calculée par une fonction de hachage. 39, 42, 73

E-ID Le terme anglais *electronic IDentity* raccourci en *e-ID* signifie en français *identité électronique*. C"est un type d'identité accordé à une entité dans un domaine numérique. ii, iii, 1, 54, 61, 67–69, 72, 111, 112

Framework C'est un ensemble d'outils et de composants logiciels qui servent à créer les fondations d'un logiciel. 56–58, 60, 74, 78–82, 84, 86–90, 98, 110

Hachage C'est une fonction informatique qui va prendre en entrée des données et générer une empreinte numérique. 39, 42–44, 73

Hypertext Markup Language Le Hypertext Markup Langauge abrégé en *HTTP* est un langage conçu pour représenter les pages Web. 5

Key Management System C'est le système qui gère les clés dans un modèle de cryptographie asymétrique. 61

Know your customer C'est un processus utilisé par les entreprises pour s'assurer que les clients sont conformes vis-à-vis des législations. Il permet également de prévenir l'usurpation d'identité, la fraude fiscale, le blanchiment d'argent ou encore le financement du terrorisme. 15

Login Le Login, en français *l'enregistrement de l'utilisateur*, est un identifiant de connexion qui permet de s'authentifier sur un système d'information pour ensuite y avoir accès. 6, 69

Man-In-The-Middle-Attack En français, l'attaque de l'homme du milieu, est une attaque informatique qui a pour but d'intercepter les communications entre deux parties, sans que l'une des deux parties ne se doute de quelque chose. Cette attaque est souvent utilisée lors de l'échange de clés asymétriques. 64

Métadonnées C'est une donnée qui permet de décrire une autre donnée. 26

Nonce Un *number used once* abrégé en *nonce* est une valeur numérique trouvée par un mineur de Bitcoin. 43

Open source Dont le code source est ouvert et peut être visualisé de tous. 13, 48, 49, 53, 55, 58, 59, 61

Password Safe c'est un logiciel qui permet à un utilisateur de gérer les différents identifiants qu'il possède.. 9

Peer-to-peer En français, pair-à-pair. C'est une technologie qui permet l'échange, sur réseau, entre deux entités sans passer par un serveur. 11, 15, 38, 55

Proof C'est une preuve de la validité d'une information. 26, 27, 92, 94, 95

Proof of Stake C'est un mécanisme de consensus où il est demandé au prouveur de prouver leur participation en prouvant qu'il possède un certain nombre de parts de la crypto-monnaie. 48

Proof of Work C'est un mécanisme de consensus cryptographique dans lequel un prouveur prouve à des vérificateurs qu'un calcul informatique a été effectué. vii, 42–45, 48, 49, 53

Public Key Infrastructure C'est une infrastructure qui permet de gérer les clés publiques dans un système de cryptographie asymétrique. 11, 47

REST Le *representational state transfer* abrégé en *REST* est une architecture logicielle qui permet de créer des services web pour la manipulation de ressources à travers un ensemble d'opérations. 56–58, 61, 80, 90, 94

Routage C'est un mécanisme qui va permettre de sélectionner des chemins dans un réseau pour acheminer une donnée d'un expéditeur jusqu'à un ou plusieurs destinataires. 45

Smart contract C'est un programme stocké sur une *Blockchain* qui s'exécute lorsque des conditions sont remplies. 34, 47–53, 65, 66

Social Engineering En français, l'ingénierie sociale est une pratique de manipulation psychologique à des fins d'escroquerie. 73

Sujet C'est l'entité sur laquelle des informations sont données. Par exemple, si sur le passeport de Monsieur Dupont il est inscrit que sa date de naissance est le 25 février 1975, le sujet de cette information est Monsieur Dupont. 25, 26, 28, 33

Swagger C'est un langage qui permet de décrire des API exprimées en JSON. 80–86, 89, 94

Ubuntu C'est un système d'exploitation. 74, 90

Uniform Resource Locator L'Uniforme Resource Locator abrégé en *URL* signifie en français *Localisateur Uniforme de Ressource*. Il représente une adresse d'une ressource unique sur le Web. 5, 34

USA PATRIOT Act La loi antiterroriste intitulée *"Uniting and Strengthening America by Providing Appropriate Tools Required to Intercept and Obstruct Terrorism Act"* traduisible en français par *"Loi pour unir et renforcer l'Amérique en fournissant les outils appropriés pour déceler et contrer le terrorisme"* donne le droit aux services de sécurité des États-Unis d'accéder aux données informatiques des particuliers et des entreprises sans avoir une autorisation et sans en informer les utilisateurs. 54

Windows C'est un système d'exploitation. 74, 90

World Wide Web L'Uniforme World Wide Web abrégé en *WWW*, il est connu en français comme *la Toile* permet de consulter des pages Web accessible sur des sites. Ce n'est qu'une des applications possible d'Internet. 5

World Wide Web Consortium Le World Wide Web Consortium est un organisme international qui développe des standards pour le web. 2, 37, 48, 49, 62

Wsl C'est une couche qui permet d'exécuter des exécutables Linux sur Windows. 74, 90

Informations sur ce travail

Informations de contact

Auteur : Valon Rexhepi

HES-SO Valais-Wallis

E-mail : *valon.rexhepi@students.hevs.ch*

Déclaration sur l'honneur

Je déclare, par ce document, que j'ai effectué le travail de bachelor ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de bachelor, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail.

Lieu, date : _____

Signature : _____